慧与国际软件人才及产业基地慧据教育系列丛书

软件测试管理及工具应用

张 爽 胡锦程 陈 飞 编著
罗 诺 审稿



清华大学出版社

慧与国际软件人才及产业基地慧据教育系列丛书

软件测试管理及工具应用

张　爽　胡锦程　陈　飞　编著

清华大学出版社
北　京

内 容 简 介

本书是面向高等专科教育、高等本科教育的软件测试管理理论加实践指导教材。由惠普软件测试专家根据多年实际项目经验编写。全书共20章，系统地介绍了软件测试管理的基本概念、软件测试的管理体系、软件测试需求管理、软件缺陷管理等一系列测试管理知识，并结合惠普软件测试管理工具ALM(Application Lifecycle Management)，介绍在实际项目中如何运用HP ALM工具进行测试管理，该工具旨在提高项目的可预测性、可重复性、改进质量以及随变更及时做出调整，通过提供对流程的系统控制，简化和组织应用程序管理，使读者能够在掌握测试管理理论知识的同时掌握在企业实际项目中整个测试的管理过程，在实际工作中能够灵活有效地开展测试管理。

本书内容丰富，结构合理，图文并茂，可操作性强，讲练结合。适合作为普通高等院校大专层次和本科层次的计算机及相关软件专业学生作为专业基础课的教材使用，同时还是测试工程师、测试经理、质量改进人员和广大软件测试爱好者的理想参考教材。

图书在版编目(CIP)数据

软件测试管理及工具应用 / 张爽，胡锦程，陈飞 编著. —北京：清华大学出版社，2014(2017.7 重印)
(慧与国际软件人才及产业基地慧据教育系列丛书)
ISBN 978-7-302-38147-1

I. ①软…　II. ①张…　②胡…　③陈…　III. ①软件－测试－高等学校－教材　IV. ①TP311.5

中国版本图书馆CIP数据核字(2014)第222418号

责任编辑：王　军　李维杰
装帧设计：牛艳敏
责任校对：曹　阳
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编：**100084
社 总 机：010-62770175　　**邮　　购：**010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京嘉实印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**26.5　　**字　　数：**662千字
版　　次：2014年10月第1版　　**印　　次：**2017年7月第2次印刷
印　　数：2701~3368
定　　价：52.00元

产品编号：061440-01

前　　言

惠普在2015年11月进行了战略拆分，正式拆分成两家独立企业，其中新成立“Hewlett Packard Enterprise (简称为 HPE)”，专注于企业级业务，核心业务就是领先的软件资产，由惠普领先的企业技术基础架构业务、软件业务和服务业务组成；另一家为HP Inc，由惠普领先的个人系统业务和打印业务组成。

2016年2月1日起，HPE正式发布了新的品牌名称及品牌LOGO。HPE中文品牌名称“慧与”应运而生。“慧”象征智慧，和“惠”同样的发音联结着我们与过去的传承与发展。“与”寓意合作，表达我们与员工、合作伙伴、客户间携手共赢的关系。

慧与国际软件人才及产业基地开展实训业务三年来，在全国几十所高校设立了共建专业，并与几千家企业建立了人才合作关系，培养了近万名软件人才。其中软件测试方向以具有省级CMA资质国际领先的软件测试中心为依托，使用慧与一流的测试工具和平台，开设了功能、性能、安全等课程，目前处于全国领先水平。

伴随着我国软件产业的蓬勃发展以及对软件质量的重视，企业对软件测试从业人员的技能要求也逐步提高，这就要求测试人员掌握目前软件测试行业最新的发展动态，运用新的测试技术、新的测试方法和新的测试工具，以满足不断前进的软件行业的要求，有效提高软件测试的效率和成果，确保软件测试的质量。

本书以掌握软件测试基础知识为起点，结合惠普测试管理工具ALM(Application Lifecycle Management)，使学生了解如何对项目的测试生命周期进行管理，帮助企业改进质量，优化测试流程，交付让用户满意的软件产品。在适当介绍理论知识，突出实践能力培养的基础上，结合惠普专家多年从事软件测试项目管理经验，编写了这本适合普通高等院校大专层次和本科层次的计算机及相关软件专业学生使用的测试管理教材。本书的目的是培养高素质、专业化的软件测试管理人才，让学生对软件测试综合管理过程有完整认识和总体把握。

全书分为3部分，共20章，第Ⅰ部分为软件测试管理基础篇(包括第1～7章)，介绍软件测试管理的定义、软件测试管理体系以及软件测试需求管理、软件测试团队管理、软件测试文档管理、软件测试缺陷管理等丰富的软件测试管理基础知识。第Ⅱ部分为HP ALM工具入门篇(包括第8～16章)，利用惠普测试管理工具ALM围绕一个真实的项目，进行发布管理、需求管理、缺陷管理等实践操作。第Ⅲ部分为HP ALM工具提高篇(包含第17～20章)，在HP ALM工具入门篇的基础上，进一步学习怎样对项目的需求进行风险分析、怎样分析和评估项目风险、怎样使用版本控制、怎样进行库管理等一系列高级内容。

软件测试管理知识近年来理论和实践都在高速发展中。书中的理论和实践都有待于在实践中不断探索和验证。欢迎广大读者对本书提出宝贵的批评和建议，以期在今后再版或改版时加以改进和完善。希望本书的面世能推动业界专家对相关专题的讨论和分析，大家一起为软件人才的培养添砖加瓦。

目　　录

第 I 部分　软件测试管理基础篇

本部分共有 7 章内容，分别是：软件测试管理概述、软件测试需求管理、软件测试团队管理、软件测试文档管理、软件缺陷管理、软件测试流程管理以及软件测试执行管理。以上是软件测试专业人才培养的基础知识，要求全面掌握和深刻理解，为此需要投入一定的学时数。

第1章　软件测试管理概述

软件测试是保证软件产品质量的最关键手段之一。软件测试管理应该贯穿于整个软件测试过程。测试管理包括测试计划、功能性测试流程、测试执行、测试用例设计、需求文档评审、缺陷管理、标准和测试过程改进、测试工具与自动化、团队建设等所有环节的管理。本章将从测试管理的角度介绍软件测试管理的意义、内涵、测试管理规范、体系和要素。作为应用型软件人才，学习完本章后，学员应具备以下能力：理解软件测试管理的重要性、主要方法与规范，并掌握怎样执行测试管理的各个重要环节。通过实际学习和实践，期望收到的效果是：

当你掌握了测试方法和流程后，对测试管理的执行将更加轻松。

当你了解了 TMM 等模型和国内外标准后，对软件测试过程的改进就会增强信心。

当你了解了测试成本后，将更加容易与其他利益相关者进行高效沟通。

当你掌握了测试管理的要素和侧重后，对软件的测试管理实践将会有明确的方向。

1.1　软件测试管理基础

美国质量保证研究所对软件测试的研究结果表明：越早发现软件中存在的问题，开发费用就越低；在编码后修改软件缺陷的成本是编码前的 10 倍，在产品交付后修改软件缺陷的成本是交付前的 10 倍；软件质量越高，软件发布后的维护费用越低。另外，根据对国际著名 IT 企业的统计，它们的软件测试费用占整个软件工程所有研发费用的 50%以上。

相比之下，中国软件企业在软件测试方面与国际水准仍存在较大差距。首先，在认识上重开发、轻测试，没有认识到软件项目的如期完成不仅取决于开发人员，更取决于测试人员；其次，在管理上随意、简单，没有建立有效、规范的软件测试管理体系；另外，缺少自动化工具的支持，大多数企业在软件测试时并没有采用软件测试管理系统。所以对国内软件企业来说，不仅要提高对软件测试的认识，同时要建立起完善的软件测试管理体系。

1.1.1　软件测试管理目标

软件测试的管理是业界专家公认的一个难题。其中有许多原因。第 1，由于软件产品的多样性和复杂性，软件的测试也是一件困难的事情。第 2，软件测试仍然没有被认为有足够的权威和专业的人才培养体系，因此找到具有资格的人专业地做测试工作并非易事。第 3，不同于所有产品开发活动被明确定义和规范化为产品设计、产品开发、质量管理等的过程，还没有为软件测试定义相似的规范化的过程。第 4，针对软件测试活动的自动化工具尚处初期阶段，需要时间使复杂的自动化工具可利用为软件测试的独立和可靠辅助工具。第 5，能准确估算软

件测试所需时间和资源的技术仍然缺乏，因此当前估算测试资源和时间的方法和技术都还是凭靠经验，是主观判断。因此测试管理并非易事，需要测试团队努力地根据客观条件积极探讨，才能找到可行的最佳实践。

既然没有人能对所有可能性进行测试，那么如何在有限的资源和条件下策划测试和执行测试管理就变得非常重要。软件测试管理的目的和结果是帮助测试团队决定最佳实践。要明确软件测试管理目标，首先要明确为什么不能测试所有可能性。

1. 不能测试所有可能性

计算机领域的先驱者 Edsger W. Dijkstra 曾说过，“测试也许可以令人信服地表明存在缺陷，但是永远无法表明不存在缺陷”。

软件由软件开发工程师编写的文本格式的源代码编译而成。但众所周知，人类的大脑会犯错，人的智慧也有局限。另外，软件开发和软件测试项目都只有有限的时间。所以，无论我们多想进行所有可能的测试，也无法全部考虑到；即使全部考虑到，也无法保证有足够长的时间来完成。此外，在大多数情况下，要进行所有可能测试的成本也太高。因为对任何程序而言，可能进行的测试数目都是无限的。下面我们来看看原因：

1) 可能进行测试的数目是无限的

用一个例子来说明我们对能够想象到的最简单的程序进行测试：该程序的功能就是在用户敲击空格键的时候在屏幕上显示内容为“你好！”的消息作为响应。为简单起见，我们要测试每次按下空格键的时候，都能得到“你好！”的消息；而每次我们按下其他键或组合键时，都不会在屏幕上得到任何响应。想想看“其他的键或组合键”会有多少测试用例？需不需要测试在按下空格键的同时再按任何其他键？当然需要！那么有多少种组合呢？

如果不能查看程序内部的代码，需要执行多少测试用例才能让你有把握说已经对所有的可能性都进行了测试呢？确定测试用例的数目，把它写下来，然后再阅读下一段，看看你的答案是否正确。

在通常情况下，测试人员无法也无须查看程序内部的代码，所以也就无法知道程序员都设置了那些特定的条件。例如，也许程序被设计成外面看起来一切正常，直到用户输入一组极不可能出现的按键序列。假设有个开发人员为了给自己工作方便留了特殊条件，他将程序设置成要求先按 W 键，接着按 3 次空格键，再按 M 键，又是按三次空格键，然后按 J 键，在输入满足这个条件时程序会在屏幕上显示：“Come On！”。你之前建立的测试用例穷举集是否能检测到这个怪异的条件？而这一条件在程序中隐藏了一条多余的而且是非预期的“Come On！”响应。

类似条件不切实际的情况的确发生过。比如有信息报道，某公司在对一个理论上应该是高度安全的应用进行技术审查时，发现一个名为 Wanda Marilyn Jones(不是真名，不过首字母一样)的程序员在编写口令保护时在软件中设置了一个以此为条件的后门。无论真正的口令是什么，她都可以绕过正常的口令保护，随时闯入系统。在严格控制下进行的一个极复杂的测试计划也未能发现这个后门，让它存在了三年，直到该公司对它进行技术审查。如果没有想到对软件进行穷举测试需要的测试数目无穷大，或者至少是无法完成的测试数目，希望这个

例子能让你理解。

另一个实际要思考的测试场景是：在不同配置下对程序进行测试的可能性，而这一问题是产品开发人员通常都会遇到的。如果程序需要在 10 种不同的 CPU 上运行，每种都分别要对应 10 种可能的内存大小和 10 种不同的磁盘驱动器大小，可能就意味着要测试 10 乘 10 乘 10，也就是 1000 种不同的配置。所以这也说明了可能要测试的可能性数目非常大。

对于很多真实的测试情况而言，这些例子可能过于简单。在真实的测试中，测试人员要解决由不同的制造商、不同的驱动程序、不同的操作系统版本、不同的语言版本、不同的浏览器、同时运行的其他程序，以及其他不同的外围设备组合带来的其他复杂性，任何一种配置都可能导致错误。要“完全”覆盖像这样的所有可能配置，会要求以 quadrillion(10 的 15 次方)计算的测试用例。而这还没有考虑对程序在任何一种配置下所应完成的各种功能进行测试。即使这一巨大数字也没有考虑到测试的执行顺序可能带来的影响。如果用户可以调用 10 个功能，仅仅使用 10 个不同的测试是不够的，因为以不同的顺序执行这些功能也许会产生不同的结果。所以我们需要 10！(10 的阶乘，大约 600 万)个测试而不只是 10 个测试才能覆盖所有的执行序列。

这些巨大的数字(所有的都乘在一起)还忽略了真正的随机性，例如某个外部设备在那一纳秒上产生中断，或者用户在那一毫秒敲击 J 键，总而言之，测试可以让人筋疲力尽，但它是不可能穷尽的。

2) 真正能执行的测试只是代表性的案例

由于我们无法测试所有的可能性，因此任何实际的测试集都是某种程度的“采样”，即以某种方式代表整个可能测试集合的一部分。当然我们希望它是最合适的代表，但这就会带来一个问题：“对谁而言是最合适的？”本质上，采样也是非唯一的结果，也是一种被认为是感性的过程。令某人满意的样本也许会让另一人觉得不满意。

那么如何决定采哪些样？我们希望知道是否获取足够大的样本来充分代表所有情况。我们也希望知道我们的确获得合适的代表性的测试用例样本。但实际上不得不承认我们的局限性。我们能做的就是在测试中尽最大努力决定最有代表性的测试用例，也就是取少量的样本，然后根据测试结果对整个软件功能或产品的质量做出判断。

3) 很难确定理想的可能测试的数目

不可能进行穷举测试导致我们要在这两个难以达到的，但是同样吸引人的目标之间进行取舍：一是我们希望测试能够覆盖所有需要测试到的条件，二是我们希望将测试集减小到可以管理和承受的程度。

要理解第一个目标的含义，不妨考虑一下有多少次测试人员并不是在寻找某种类型的缺陷，却碰巧发现了某些很严重的缺陷。其实可能他们发现这些缺陷只是运气好，而不是由于他们正在执行一组精心设计的用于发现这个特定问题的测试。缺陷就这样被偶然发现了。也许只是碰巧而已，但我们希望测试用例的设计和专业化的执行能有很到位的测试覆盖面，从而能不漏掉所有的严重缺陷。

4) 用较少的测试资源获取更多的信息

将测试集减小到可以管理和承受的程度通常是测试团队追求的目标。不少测试团队被要求或命令要在测试队伍更小而责任更大的条件下生存甚至是发展。业界有人讲过一种比较极端的情况：有名测试人员需要处理如下很为难的局面："我们的测试团队刚刚从 30 人减少到 3 人，而且还要"确保"这个产品的质量。我如何确定该测试哪些东西？"

有一种观点认为测试人员不能"确保"任何事，所以他们根本就不应该进行尝试。但是这样的观点无法说服任何一名正在艰难的经济条件下努力保持公司运行的管理人员。那么该怎么办？无可否认，减小后的团队无法完成更大的团队本来可以完成的所有事，但是可以在他们可能进行的测试中加以选择，可以找出让有限的资源得到最佳利用的那些测试。

顾问给出的建议提供了很好的基础："首先，要认识到任何测试集都是一种采样方法。然后，无论你有多少资源，都要尽可能选择那些具有最强代表性的测试集。"

2. 软件测试管理的目标

业界有不少专家认为软件测试的终极目标是：有效、全方位提高测试覆盖率的手段。那么软件测试管理的目标就应该是通过系统、高效、适用的技术、方法和体系来监督、促进和达到软件测试的这个目标。决定软件测试管理的目标时应记住考虑以下几方面：

1) 可用测试资源

从其他的课程已经了解到软件的可用测试资源有很多，包括硬件和软件环境，测试工具等系统资源以及人力资源。在这之中，最重要的是人力资源，包括测试项目负责人、测试分析员、测试设计员、测试程序员、测试员、测试系统管理者以及配置管理员等。此外要明确测试范围，以估算测试需要的资源和可用的资源。

2) 使用适当的测试技术和方法

软件测试有很多方法，但是不同类型的软件和需求是决定使用什么和怎样使用测试方法的前提。根据不同的软件测试需求和场景，正确运用软件测试的技术和方法是成为一名成功的测试管理者的前提。测试管理的重要目的之一就是决定使用适当的测试技术和方法。

3) 明确具体软件测试任务

具体测试任务有很多，做好测试管理当然要明确测试任务，也就是明确测试范围。用流程图或表格列出所有主要测试任务是很好的测试管理办法。图 1-1 所示为软件测试任务图。

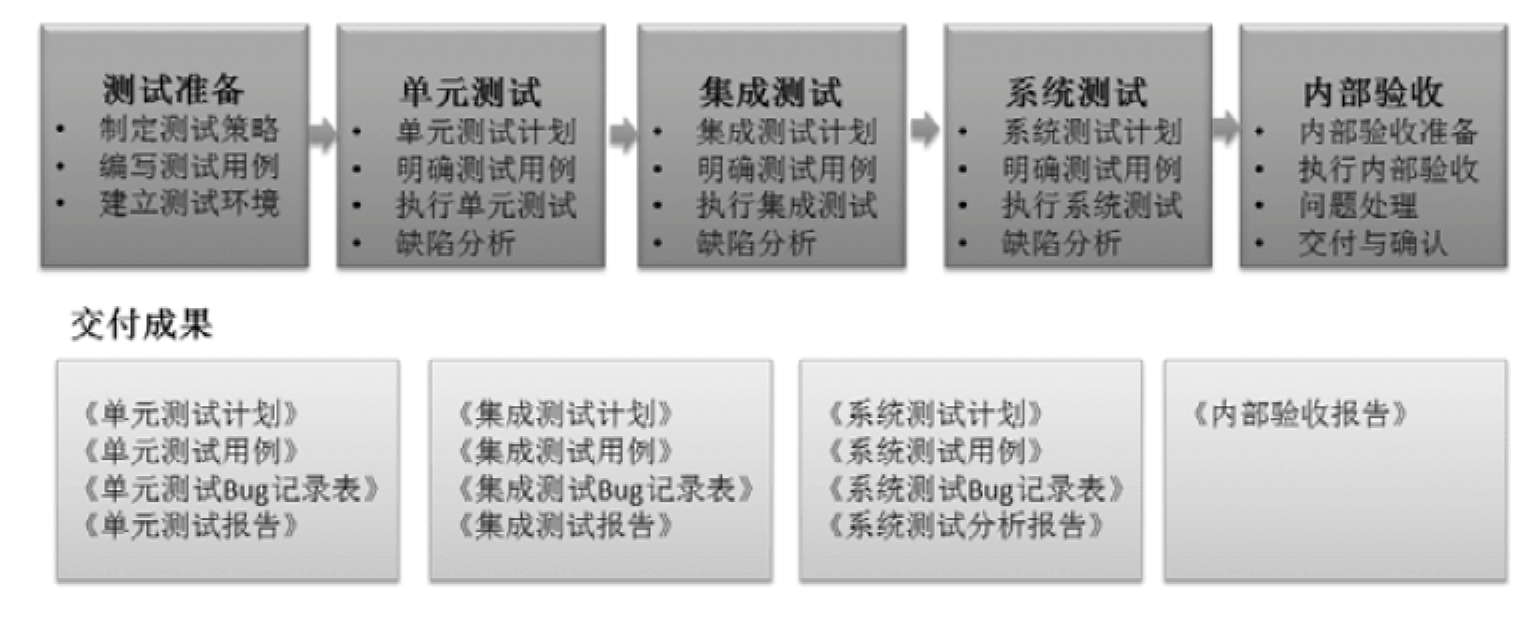


图 1-1　软件测试任务图

1.1.2　定义与分类

软件测试的管理还没有业界统一且公认的定义和分类，本节就常见的测试管理定义和分类加以分析。

1. 软件测试管理定义

对软件测试管理的定义和目的可以有不同的理解，但总体都是关于对每项具体软件测试活动以及总体软件测试全局的监督、评估、决策和管理的过程。因而进行软件测试的管理就是对每一种具体测试任务、流程、体系、结果、工具等进行具体监督和管理。

尽管很多业界专家对软件测试管理有不同的定义，但无论软件测试管理怎样定义，软件测试管理是艺术，是责任，同时还是科学。

2. 测试管理分类补充要点说明

下面讨论常见的分类以及几项需要特别说明的测试管理重点类别，并从测试管理的角度加以说明。常见的实践是可以把软件测试管理分为 8 类。下面逐一分析：

- 软件测试需求管理
- 软件测试质量管理
- 软件测试团队管理
- 软件测试文档管理
- 软件测试缺陷管理
- 软件测试环境管理
- 软件测试流程管理
- 软件测试执行管理

除以上 8 项类别外，测试管理应考虑的其他内容和范围也很重要。在讨论范围与来源时会讨论更详细的内容。在此对几个专项测试管理做一下特别说明：

1) 测试计划的管理

测试计划的制订是必需的测试环节，本书没有单独分析，是因为测试计划是一种测试文档，因此放在测试文档管理部分介绍。测试计划和其他测试文档一样，应作为一类重要的测试管理任务进行全程状态跟踪，也需要有流程来进行管理。

2) 测试用例管理

编写测试用例和测试缺陷报告在本书被认为是测试文档的一种，详见第 4 章。

3) 测试报告管理

测试报告是测试管理系统存储库中的一类受控对象，作为受控对象与测试计划双向关联。测试报告管理也包括在第 4 章中。

4) 测试成本管理

测试成本管理的基础是测试资源估算，包括人力资源、测试需求和范围估算，以及测试完成所需时间、风险等估算，这些都需要经验丰富的资深测试人员或测试经理等才能操作，

本书面向的对象是高校学生和初级软件工程师，在此没有作为特别讨论的专题。

5) 测试风险管理

软件测试风险是不可避免并总是存在的，所以对测试风险的管理非常重要，必须尽力降低测试中存在的风险，最大程度保证软件质量和满足客户的需求。软件测试的风险分析常包括在软件测试计划文档中。

1.1.3　范围与来源

软件测试的管理也就是对软件测试项目的管理，因此软件测试管理知识和项目测试实践经验需要结合。测试管理的内容和范围很多，本章以及后续的章节将会以一个实际的项目案例为基础，对测试管理中涉及的各种测试文档、技术和方法、管理技能、工具和自动化等进行详细阐述。

测试管理应该贯穿于整个测试过程，测试经理和其他测试负责人需要详细了解测试过程中的各项测试活动，并对它们进行有效管理。

1. 常见的测试管理范围及内容

1) 测试管理文档：描述与 IEEE Std 829–2008 定义兼容的测试计划、测试设计规格说明、测试用例规格说明和测试规程规格说明等文档的内容；同时还阐述测试策略中包含的主要元素，以及如何对与测试策略之间存在的偏差进行监控和管理。

2) 明确测试范围：根据测试需求和目标，确定测试范围、责任分工、交付标准等。

3) 测试资源估算：考虑影响测试估算的各个因素。通过具体的项目案例详细地分析各种可行的测试估算技术。例如，基于团队的测试估算、基于百分比的测试估算、基于测试规模的测试估算等。决定测试团队需要的资源和可用资源、测试人员技能要求等。

4) 测试计划制订：参照测试计划的文档模板，以及如何根据组织结构、产品风险、项目风险、产品规模和类型等因素，并参考 IEEE Std 829-2008 中建议的测试计划结构或其他最佳实践适用的测试计划，从项目风险、产品风险、测试资源计划和分配、测试优先级等方面制订测试计划。

5) 测试过程监控：分别从风险、测试、覆盖率、缺陷和信心 5 个方面，阐述测试过程的监控活动，并讲解如何利用在监控活动中观察到的测试过程相关的信息和问题，制订应对计划控制当前的测试过程，并提供改进建议。

6) 缺陷会审和流程：从测试管理角度，对于发现的缺陷，特别是严重程度高的缺陷，怎样与开发人员、项目经理、客户和设计人员一起通过缺陷会审决定是否修复。

7) 确定测试技术和方法：根据具体项目，决定用哪些测试技术和方法、怎样执行以及谁负责哪部分等。

8) 基于风险的测试：阐述风险的基本概念和风险管理的基本过程。测试风险分为产品风险和项目风险，测试计划的制订应该基于对测试风险的分析。测试经理需要汇总项目利益相关者对项目风险的不同观点，使用汇总后的结果开展减轻风险的测试活动。测试经理应该根据测试风险的状态，分析和报告测试结果，确定剩余的风险以帮助项目管理人员做出正确的

决策。

9) 失效模式和影响分析：明确失效模式、影响和危急程度的基本概念和处理办法。

10) 测试管理系统和工具：针对不同的测试活动和测试对象(例如探索性测试、综合系统测试和安全关键系统测试等)实现测试的自动化可用的辅助工具，决定测试工具和缺陷管理系统等。

下面以单元测试策略和集成测试策略为例，说明具体测试策略常见的更具体的内容和范围。测试策略的管理要确认这些规范的落实，并和相关团队沟通。遇到问题及时处理和纠正，以确保测试项目的继续推进，并能按质量、时间、预算、期望结果完成项目。

2. 单元测试策略

单元测试(Unit Testing)又称模块测试，是针对软件设计的最小单位——程序模块进行正确性检验的测试工作。其目的在于发现各模块内部可能存在的各种差错。

1) 测试对象

单元测试集中对用源代码实现的每一个程序单元进行测试，检查各个程序模块是否正确实现了规定的功能。只测单元的内部行为，单元间接口不在此时的单元测试活动中，软件的独立单元将在与程序的其他部分相隔离的情况下进行测试。

2) 测试目标

- 分别完成每个单元的测试任务，以确保每个模块能正常工作。
- 测试健壮性、效率和可维护性。

3) 测试策略

单元测试需要从程序的内部结构出发设计测试用例，多采用“白盒”测试技术，黑盒测试为辅。多个模块可以平行地独立进行单元测试。把单元测试当作纯粹的“黑盒”测试是错误的。

4) 单元测试的考虑

- 模块接口
- 算法和逻辑
- 数据结构(全局和局部)
- 边界条件
- 独立的路径
- 错误处理

5) 单元测试的步骤

一般认为，单元测试的时机是在源程序编制完成并通过静态分析、代码复审和编译检查后，便可开始单元测试。模块并不是独立的程序，在考虑测试模块时，同时要考虑它和外界的联系，应为测试模块开发一个驱动模块(driver)和(或)若干个桩模块(stub)。

驱动模块和桩模块是测试使用的软件，而不是软件产品的组成部分，但需要一定的开发

费用。因此开发驱动模块和桩模块应尽量简单。提高模块的内聚度可简化单元测试。如果每个模块只完成一个功能，所需测试用例数目将显著减少，模块中的错误也更易发现。如果一个模块要完成多种功能，可以将这个模块看成由几个小程序组成。必须对其中的每个小程序先进行单元测试工作，对关键模块还要做性能测试。

遗憾的是，很多程序单元用简单的驱动模块和桩模块无法完成测试，这些模块的单元测试只能采用一些综合测试方法(常放到集成测试中进行)。

6) 单元测试的执行

- 检查编码是否遵循软件编程规范和标准
- 自动或手动分析程序
- 设计测试用例并运行
- 错误跟踪分析

在单元测试的同时可进行组件和模块测试，发现并排除在模块连接中可能出现的问题，最终构成符合要求的软件系统。

3. 集成测试策略

1) 概述

集成测试(Integration Testing)也称为组装测试或联合测试，是在单元测试的基础上，将模块按照设计要求组装起来进行测试，主要目标是发现与接口有关的问题。子系统的组装测试称为部件测试，所做的工作是找出组装后的子系统与系统需求规格说明之间的不一致。

2) 集成测试的目的

进行集成测试主要有以下考虑：

- 一个模块可能对另一个模块产生不利的影响。
- 将子功能合成时不一定产生所期望的主功能。
- 独立可接受的误差在组装后可能会超过可接受的误差限度。
- 可能会发现单元测试中未发现的接口方面的错误。
- 在单元测试中无法发现的时序问题(实时系统)。
- 在单元测试中无法发现的资源竞争问题。

鉴于此，集成测试的目的可以归结为：

- 在模块组装后查找模块间接口的错误。
- 模块间的数据交换是否丢失、有错(通信、数据内容、时序等)。
- 各个子功能组合起来之后能否达到预期要求的父功能。
- 全局数据结构是否有问题。
- 单个模块的误差累积是否会放大，从而达到无法接受的程度。

3) 集成测试的方式

通常，把模块组装成为系统的方式有两种：一次性组装方式和增殖式组装方式。

1.1.4　软件测试管理特色

测试管理的最佳实践必须根据实际待测软件团队的条件、状况、可用资源等很多因素决定。此外还必须根据软件研发不同的模式制定流程、训练测试人员和测试管理办法、体系、工具等。本节在讨论一般测试管理特色的基础上，特别以敏捷开发模式为例讨论敏捷测试管理特色。

1. 一般测试管理特色

为做好测试管理工作，需要注意以下几点：

1）学习和推广最佳实践

学习企业内、团队内、业界最佳实践。特别是最新可用的测试工具、模板、流程等任何可以帮助团队提高测试效率的方式。

2）考虑总体开发项目

测试管理的度量面向的不仅仅是测试过程的改进，测试效果的加强，面向的是整个开发过程，并始终将质量监督放在工作首位。

3）建立度量数据库

质量第一，质量需要度量，从而对收集的数据、分析的方式及结果进行完整、规范保存。这个数据库面向的是软件开发过程的持续改进，其中的数据是可复用的，可供多个项目参考使用，不随当前项目的结束而消失，而会作为历史信息持续保存，从而为测试及其他软件过程的改进提供更客观、更全面的度量数据。

4）关注过程的改进

度量过程是为了对测试及其他软件过程的改进提供参考依据，其自身运作方式的合理性会直接影响度量结果的准确性。

2. 敏捷测试管理特色

本系列书籍中已做过关于不同软件开发模式的介绍，有传统的瀑布式、螺旋式、敏捷开发等。软件测试管理的方法对于不同的开发模式也有所不同。以敏捷开发模式为基础的敏捷测试管理有自己明显的特点。

1）敏捷开发的流程

为做好敏捷测试管理，要注意敏捷开发的流程。图 1-2 所示为敏捷开发迭代流程图。

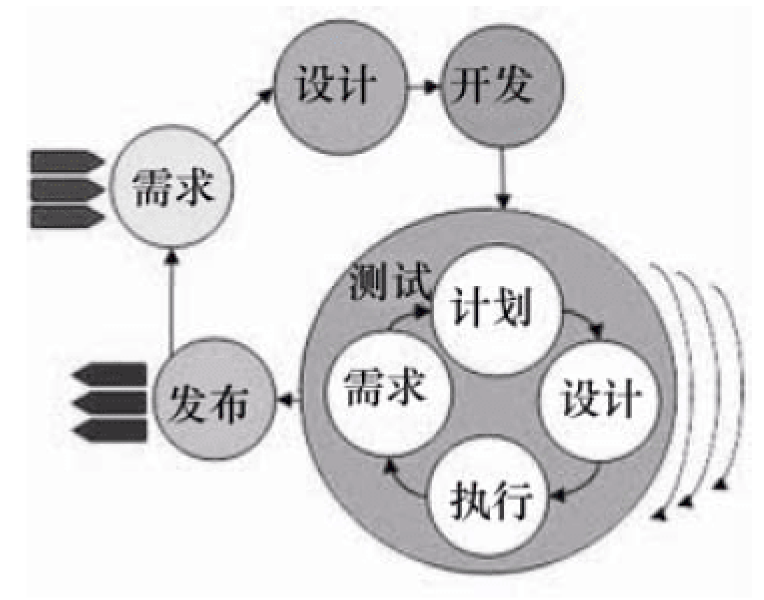


图 1-2　敏捷开发迭代流程图

根据敏捷开发流程图，敏捷测试应该是适应敏捷方法而采用的新的测试流程、方法和实践，对传统的测试流程有所剪裁，有不同的侧重，例如减少测试计划、测试用例设计等工作的比重，增加与产品设计人员、开发人员的交流和协作。在敏捷测试

流程中，参与单元测试，关注持续迭代的新功能，针对这些新功能进行足够的验收测试，而对原有功能的回归测试则依赖于自动化测试。

针对敏捷软件开发模式的说明，我们可以看出敏捷测试不能简单地理解为测试更快，绝对不是以相对以前更少的时间进行测试，也不是将测试的范围缩小或将质量降低来减少测试任务。如果将过去传统的测试流程和方法硬塞入敏捷开发流程中，测试工作可能会事倍功半，测试人员可能会天天加班，但却发挥不了应有的作用。

2) 质量问题持续反馈

同时由于敏捷方法中迭代周期短，测试人员尽早开始测试，包括及时对需求、开发设计进行评审，更重要的是能够及时、持续地对软件产品质量进行反馈。简单地说，敏捷测试管理要特别注意的就是持续地对软件质量问题进行及时反馈，如图 1-3 所示。

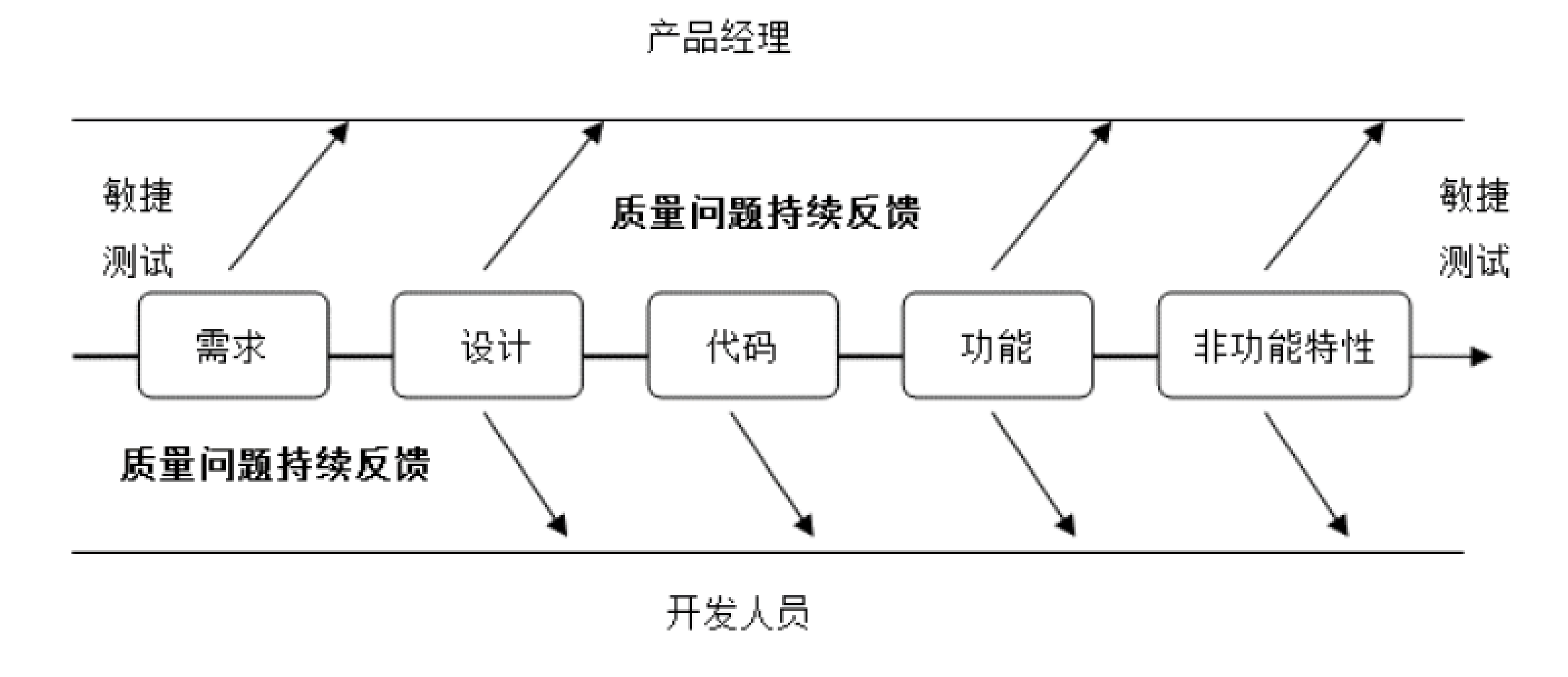


图 1-3　敏捷测试管理=质量问题持续反馈

3) 自动化测试策略

由于开发周期短，需求、设计等方面沟通也需要花费很多时间，没有足够时间开发自动化测试脚本，至少对新功能的测试很难实现自动化测试。这时候，就需要正确的策略来提高自动化测试的效益，如图 1-4 所示。

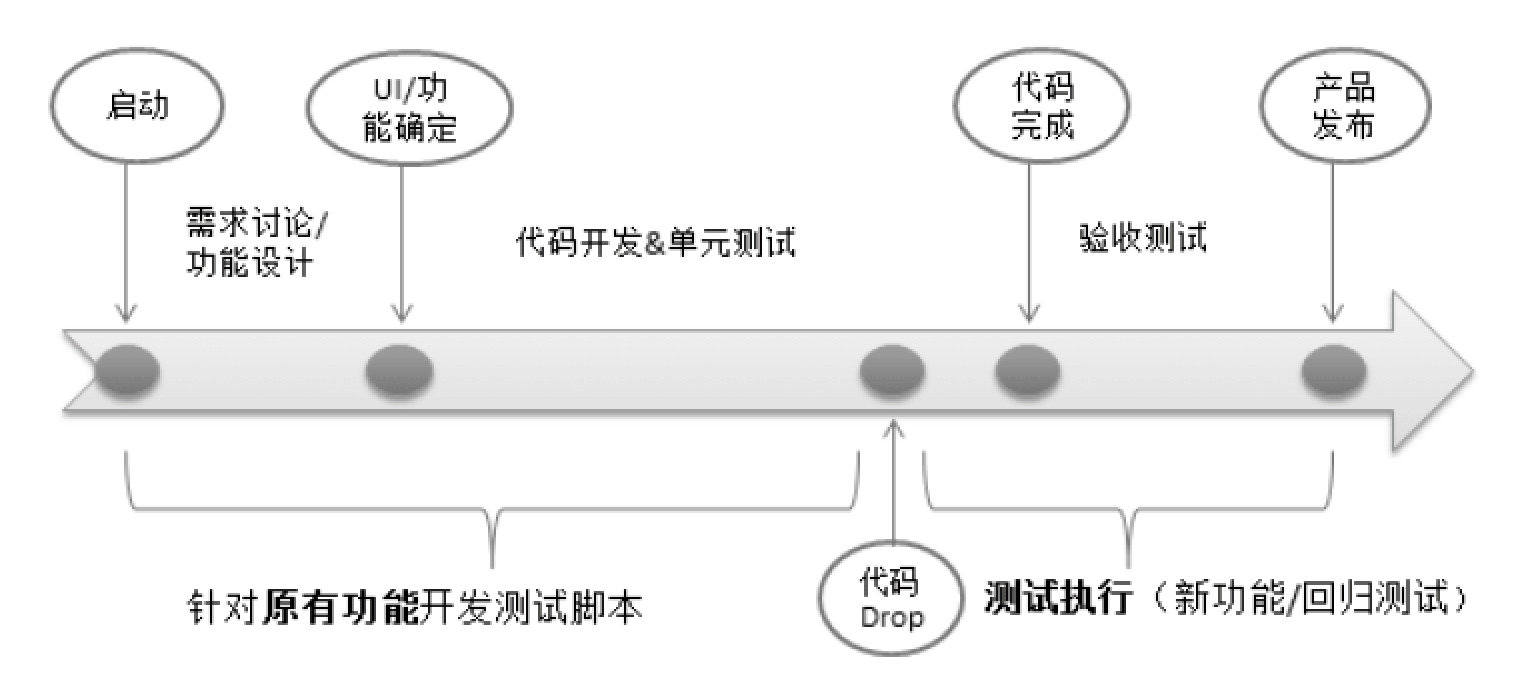


图 1-4　自动化测试策略

敏捷测试管理要制定的敏捷自动化测试策略应注意做好以下 6 方面：

(1) 构建灵活、开放的自动化测试框架，如基于关键字驱动的自动化框架，使测试脚本

的开发简单易行，方便脚本维护。

(2) 针对稳定的产品特性开发自动化测试脚本，也就是针对前期完成的已有功能开发自动化测试脚本，而对大部分新功能的测试采用手工测试。

(3) 集中精力在单元层次上实现自动化测试，主要由开发人员实施，测试人员提供单元测试框架，并辅助完成一些所需的基础工作。

(4) 在产品设计、编程时就很好地考虑自动化测试的需求，使全面、自动化的底层测试、接口测试成为可能，尽量避免用户界面的自动化测试。

(5) 良好的 IT 基础设施，包括自动化构建软件包、自动化版本验证(BVT)、自动化部署、覆盖率自动产生等。

(6) 选择合适高效的敏捷测试工具。

4) 敏捷测试管理工具

自动化测试依赖于测试工具，好在已有很多敏捷测试工具。由于篇幅受限，这里只是简单地列出一些常用的敏捷测试工具：敏捷测试过程管理工具有 HP Agile Manager 和微软的 Visual Studio 2012，包括 TFS 2012、Scrum 模板(MS VS Scrum 1.0)、Test Manager 2012、Coded UI Test 等。

敏捷测试管理的特色就是体现“敏捷”开发模式的不同特色：敏捷测试就是持续测试、持续反馈，扮演“用户代表”角色，确保产品满足客户的需求。

敏捷功能测试 = 新特性的手工测试(用例验证和探索性测试) + 原有功能的自动化测试(回归测试)。

同时敏捷测试人员和开发人员的区别越来越小，理想情况下，敏捷方法中，测试人员和开发人员在不同的迭代周期可以互换。也就是说，开发人员能做测试，测试人员也懂编程。可以看懂代码，也可以写代码。

1.2　软件测试管理体系

建立软件测试管理体系的主要目的是确保软件测试在软件质量保证中发挥应有的关键作用。特别体现在以下 5 方面：

1. 对软件产品的评估和测量

对软件产品的特性进行评估和测量，主要依据软件需求规格说明书，验证产品是否满足要求。所开发的软件产品是否可以交付，要预先设定质量指标，并进行测试，只有符合预先设定的指标，才可以交付。

2. 对软件产品的缺陷识别和控制

对软件测试中发现的软件缺陷，要认真记录它们的属性和处理措施，并进行跟踪，直至最终解决。在排除软件缺陷之后，需再次进行验证。直到认为符合软件质量发布标准时方可发布，即投入市场为用户所用。

3. 产品设计和开发的验证

通过设计测试用例，对软件功能的需求分析、软件设计、程序代码进行验证，确保程序代码与软件设计说明书一致，以及软件设计说明书与需求规格说明书一致。对于验证中发现的不合格现象，要认真记录和处理，并跟踪解决。解决之后，也要再次进行验证。

4. 软件过程的监视和测量

从软件测试中可以获取大量关于软件过程及结果的数据和信息，它们可用于判断这些过程的有效性，为软件过程的正常运行和持续改进提供决策依据。

5. 有流程和规范指导

本书侧重介绍和分析三部分内容：ISO 9000 国际质量管理和质量保证标准、软件测试成熟度模型和如何建立测试管理体系。

1.2.1　软件测试成熟度模型(TMM)

作为一类等级递增模型，TMM 体现了 20 世纪 50 年代到 20 世纪末的测试阶段划分和测试目标定义的发展历程。TMM 源于当前的工程实践总结，并充分利用了 Beizer 的测试人员思考模式的进化模型。能够用于分析软件测试机构运作过程中最优秀或最混乱的区域，并辅助软件测试机构进行测试过程的评估与改进。TMM 是当前影响力最大的软件测试过程模型，具有如下优点：

- 等级水平结构、关键活动和角色的定义最为精细；
- 测试相关因素覆盖最全面；
- 支持测试过程成熟度增长；
- 有定义良好的评估模型的支持；
- 实施 TMM 能改进测试过程，并有助于提高软件质量、软件工程生产力和缩短研发周期，减少投入。

TMM——软件测试能力成熟度模型有 5 级(参见表 1-1)。

表 1-1　软件测试能力成熟度模型的等级表

成熟度等级[i]	名　称	描　述
TL[1]	初始级	测试是一个混乱的过程，没有被很好地定义，缺乏测试工具，测试人员也没有被培训，并且没有和调试区别开
TL[2]	阶段定义级	测试被定义为紧接着编码后的一个阶段，测试过程有了一定程度的标准化，基本的测试技术和方法得到适当使用
TL[3]	集成级	测试的目标基于系统的需求，并且有正式的测试组织。具有正式的测试技术培训，可控制和监控测试过程，并且考虑使用自动化测试工具。测试计划的制订从需求分析阶段开始，并且贯穿整个生命周期
TL[4]	管理和测量级	测试成为被度量并量化的过程，测试用例被收集并且被保存

		在测试数据库中，测试过程中发现的缺陷被登记，评审被作为测试和质量控制活动
TL[5]	优化级	测试不是行为，而是一种自觉的约束

1.2.2　如何建立测试管理体系？

所谓“工欲善其事，必先利其器”，有了事半功倍的工具，自然能提高工作效率，软件测试管理系统就是建立软件测试管理体系、保证软件测试顺利进行的利器。测试管理体系既可以通过功能强大的测试管理辅助工具来体现，也可以通过一组流程等来体现。下面介绍主要环节，并以实例来说明几种测试管理体系的架构和主要功能。

1. 应用过程方法和系统方法来建立软件测试管理体系

一般应用过程方法和系统方法来建立软件测试管理体系，也就是把测试管理作为一个系统，对组成这个系统的各个过程加以识别和管理，以实现设定的系统目标。同时要使这些过程协同作用、互相促进，从而使它们的总体作用大于各过程作用之和。主要目标是在设定条件的限制下，尽可能发现和排除软件缺陷。比如测试管理系统主要由下面 6 个相互关联、相互作用的过程组成：

1) 测试规划

确定各测试阶段的目标和策略。这个过程将输出测试计划，明确要完成的测试活动，评估完成活动所需要的时间和资源，设计测试组织和岗位职权，进行活动安排和资源分配，安排跟踪和控制测试过程的活动。

测试规划与软件开发活动同步进行。在需求分析阶段，要完成验收测试计划，并与需求规格说明一起提交评审。类似地，在概要设计阶段，要完成和评审系统测试计划；在详细设计阶段，要完成和评审集成测试计划；在编码实现阶段，要完成和评审单元测试计划。对于测试计划的修订部分，需要进行重新评审。

2) 测试设计

根据测试计划设计测试方案。测试设计过程输出的是各测试阶段使用的测试用例。测试设计也与软件开发活动同步进行，其结果可以作为各阶段测试计划的附件提交评审。测试设计的另一项内容是回归测试设计，即确定回归测试的用例集。对于测试用例的修订部分，也要求进行重新评审。

3) 测试实施

使用测试用例运行程序，将获得的运行结果与预期结果进行比较和分析，记录、跟踪和管理软件缺陷，最终得到测试报告。

4) 配置管理

配置管理是软件配置管理的子集，作用于测试的各个阶段。管理对象包括测试计划、测试方案(用例)、测试版本、测试工具及环境、测试结果等。

5) 资源管理

包括对人力资源和工作场所，以及相关设施和技术支持的管理。如果建立了测试实验室，那么还存在其他的管理问题。

6) 测试管理

采用适宜的方法对上述过程及结果进行监视，并在适当时机进行测量，以保证上述过程的有效性。如果没有实现预定结果，则应进行适当调整或纠正。

此外，测试系统与软件修改过程是相互关联、相互作用的。测试系统的输出(软件缺陷报告)是软件修改的输入。反过来，软件修改的输出(新的测试版本)又成为测试系统的输入。

根据上述过程，可以确定建立软件测试管理体系的 6 个步骤：

(1) 识别软件测试所需的过程及应用，即测试规划、测试设计、测试实施、配置管理、资源管理和测试管理。

(2) 确定这些过程的顺序和相互作用，前一过程的输出是后一过程的输入。其中，配置管理和资源管理是这些过程的支持性过程，测试管理则对其他测试过程进行监视、测试和管理。

(3) 确定这些过程所需的准则和方法，一般应制定这些过程形成文件的程序，以及监视、测量和控制的准则和方法。

(4) 确保可以获得必要的资源和信息，以支持这些过程的运行和对它们进行监测。

(5) 监视、测量和分析这些过程。

(6) 实施必要的改进措施。

2. 测试管理辅助工具

图 1-5 为惠普软件测试管理工具 Application Lifecycle Management 的登录界面(以下简称 HP ALM)。



图 1-5　惠普软件测试管理工具的登录界面

测试管理工具用于对测试进行管理。一般来说，测试管理工具对测试需求、测试计划、测试用例、测试实施进行管理，并且测试管理工具还包括对缺陷的跟踪管理。测试管理工具能让测试人员、开发人员或其他的 IT 人员通过中央数据仓库，在不同地方能交互信息。测试管理工具将测试过程流水化，从测试需求管理到测试计划、测试日程安排、测试执行，最后到出错后的错误跟踪，实现了全过程的自动化管理。

测试管理工具的代表有 HP Application Lifecycle Management 等，如图 1-5 和图 1-6 所示。可扩展的统一平台的目标是自动化交付安全、可靠且高质的应用，能让你为需求管理、测试管理、缺陷管理和业务组件实施完整的 IT 质量管理基础设施，确立一致的可重复流程并应用最佳实践。 此外，惠普敏捷管理器加载项能让你管理敏捷测试，HP Sprinter 和惠普统一功能测试分别可让你进行综合、极具创新的手动和自动测试。

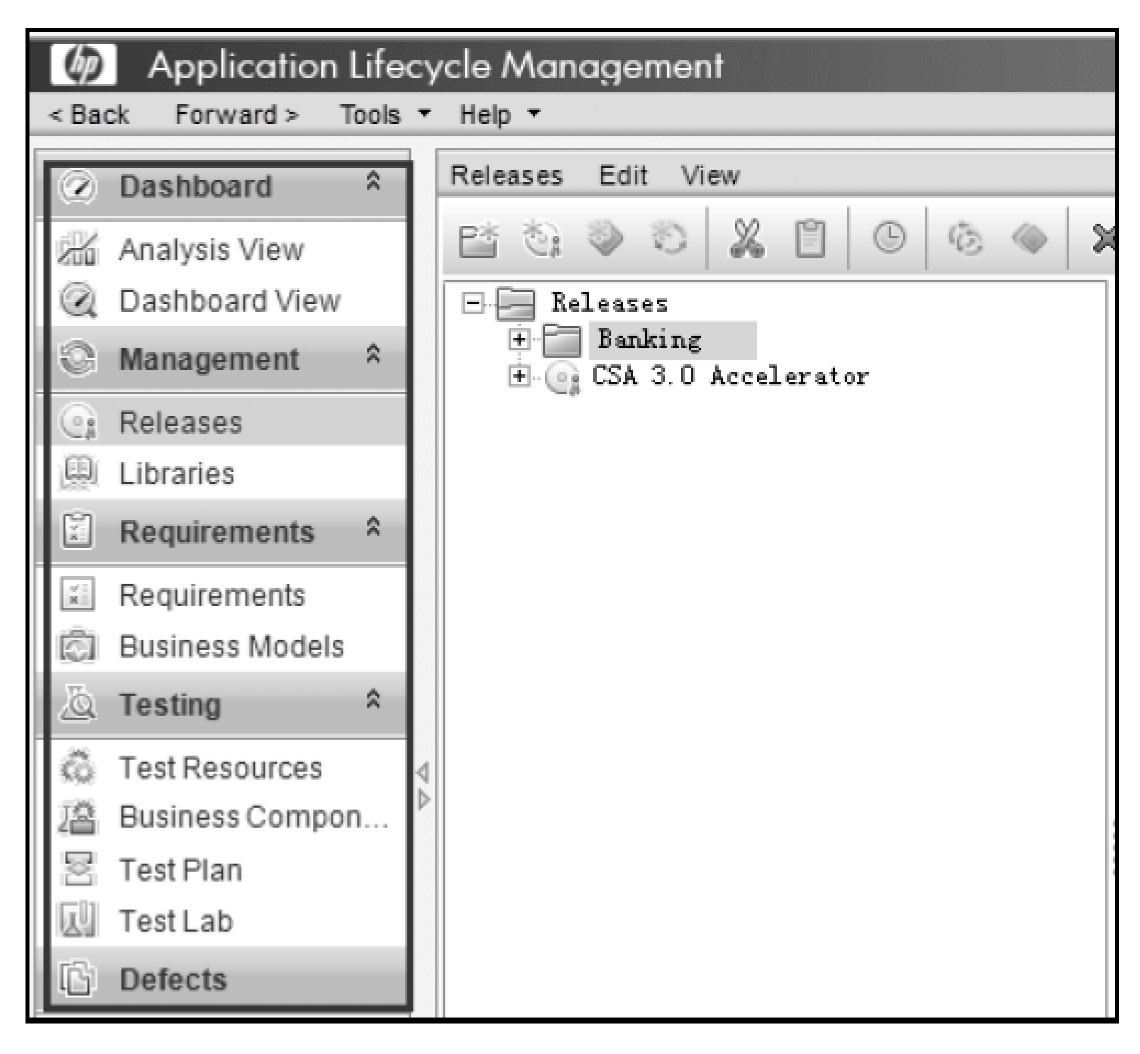


图 1-6　惠普软件测试管理工具的界面

3. 测试管理辅助工具的架构

功能强大的测试管理辅助工具能提供很多功能，如测试需求管理、测试用例管理、测试业务组件管理、测试计划管理、测试执行、测试结果日志查看、测试结果分析、缺陷管理，并且支持测试需求和测试用例之间的关联关系，可以通过测试需求索引测试用例等。

下面以某机构研发的软件测试管理平台架构图为例说明如何设计软件测试管理平台的常见架构和要素，如图 1-7 所示。

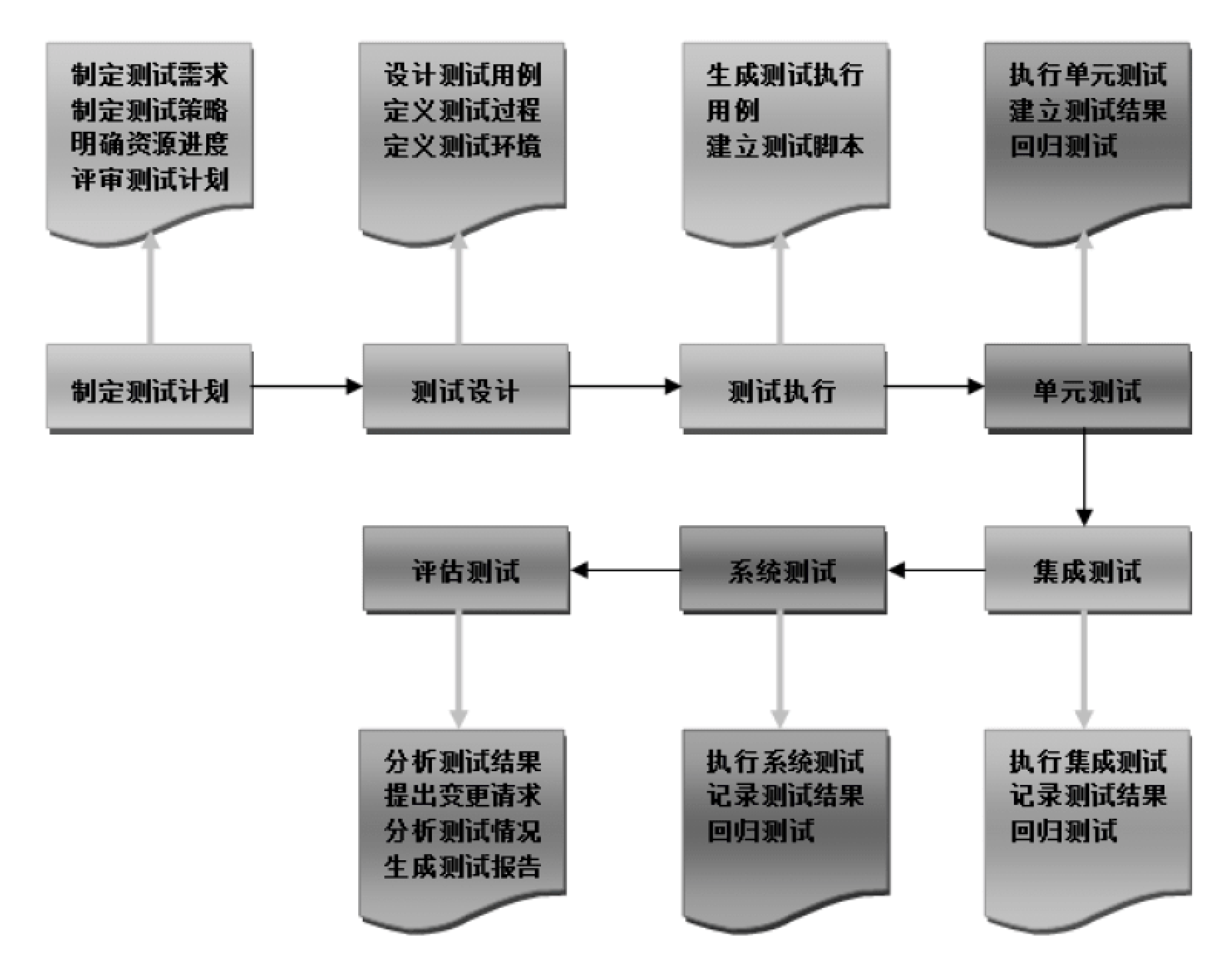


图 1-7　某测试管理体系的架构

4. 软件测试管理平台实例分析

图 1-8 是某企业用于测试管理的结构图，体现了相关环节的顺序，能帮助测试团队理解测试管理的范围和主要环节。

该测试管理平台系统专注于测试流程的管理，管理功能全面，对测试流程的设计科学、规范、合理。系统的开发是在充分借鉴国际知名大公司在测试领域尤其是测试流程管理方面的经验，参考国外知名测试管理软件，并结合开发人员在业界的经验和对国内软件开发现状的把握等基础上开发而成，非常贴近国内用户的需求；具有测试计划、测试用例、测试方法的编辑和管理功能，问题(缺陷)的跟踪处理功能，自动生成流程中各个活动的 Word 文档的功能，同时具有测试统计分析、决策支持能力，为用户提供有价值的数据挖掘服务，方便经济的技术支持和服务，技术实现上采用 B/S 和 C/S 混合结构，使得使用和维护更为方便。

1) 技术特点

(1) 技术实现上采用 B/S 和 C/S 混合结构，系统设计采用三层结构，提高了系统灵活性；

(2) 遵循国家标准《GB/T15481-2000 检测和校准实验室能力的通用要求》，对软件检测的全过程进行管理；

(3) 提供测试流程建模和测试人员角色权限的管理功能，为软件检测过程工程化和规范化提供保障；

(4) 网上受理检测业务和软件缺陷发送追踪管理的设计，体现“客户为中心”的思想；

(5) 软件测试方法模板和测试用例库的设计，为累积测试经验、支持测试知识共享和促进组织学习创造了条件；

(6) 测试文档自动生成、导出和测试结果分析的设计，提高了测试工作的效率和质量；

(7) 在测试文档、测试规范和测试用例等方面贯彻软件复用的思想，提高了测试工作的有效性。

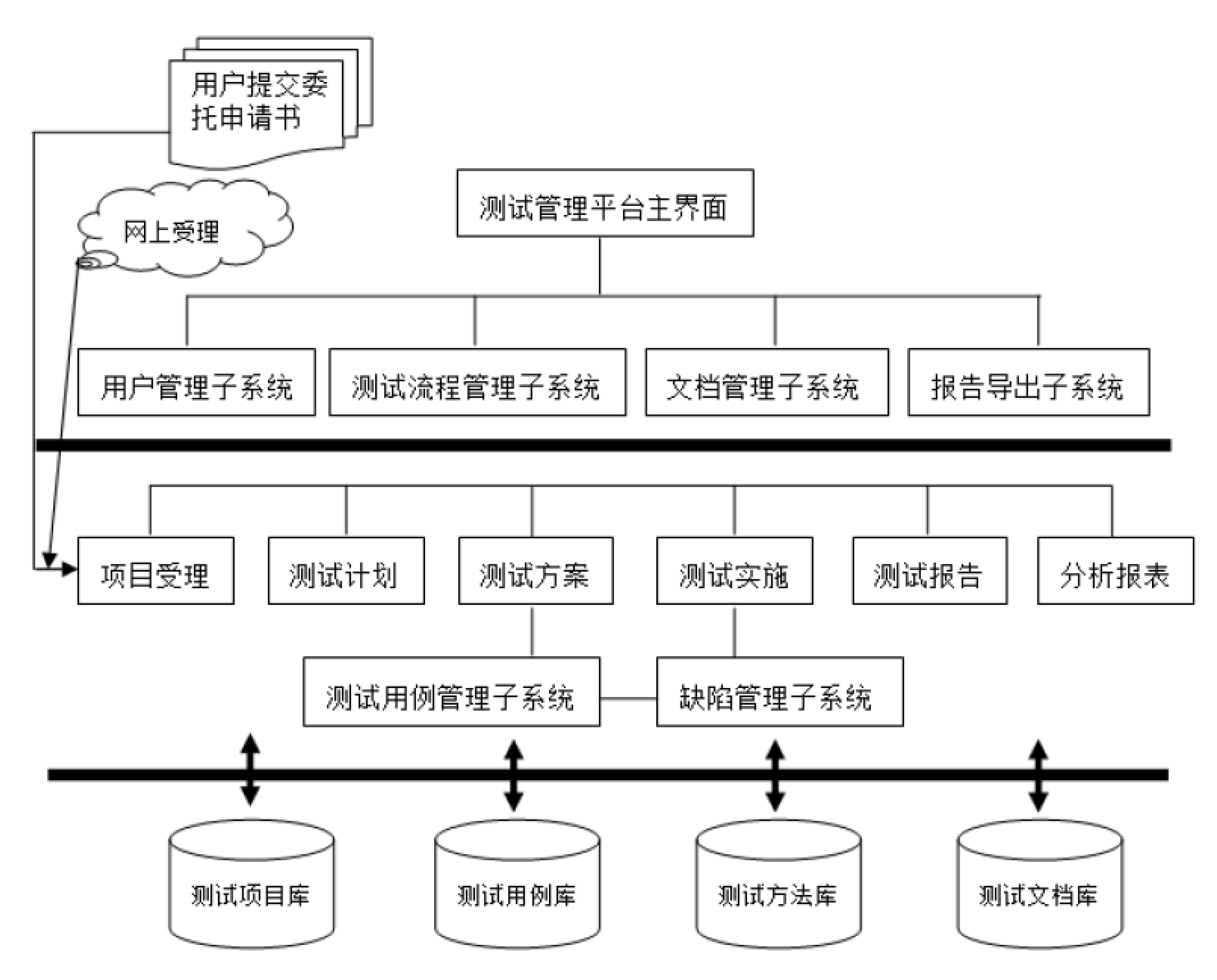


图 1-8　某软件测试管理平台的结构图

2) 功能架构

平台功能和架构由 9 个子系统组成：

(1) 测试流程管理子系统

(2) 测试方案(细则)辅助生成子系统

(3) 测试用例库管理子系统

(4) 测试文档自动生成子系统

(5) 用户权限管理子系统

(6) 统计和分析子系统

(7) 文档管理子系统

(8) 测试项目网上受理子系统

(9) 软件缺陷跟踪子系统

本平台采用三层结构设计，系统分为两部分：一部分属于 Web/Browser，用于来自 Internet 网络用户的委托软件测试项目的受理。另一部分属于 Client/Sever，用于上门测试用户的委托软件测试项目的受理，并对两部分受理的项目进行测试计划的撰写，测试方案和用例的建立，测试用例的执行，测试工具的选择，测试的执行，测试的跟踪和测试文档的自动生成。

3) 平台的主要特色

系统遵循国家标准《GB/T15481-2000 检测和校准实验室能力的通用要求》。对软件检测

的全过程进行管理，将测试工程化的基本思想结合到测试的操作流程中，可以把软件测试的过程分解为：项目受理、测试计划、测试方案、测试实施、测试报告和测试项目归档，共 6 个子过程，每个子过程对应的具体活动如图 1-9 所示。

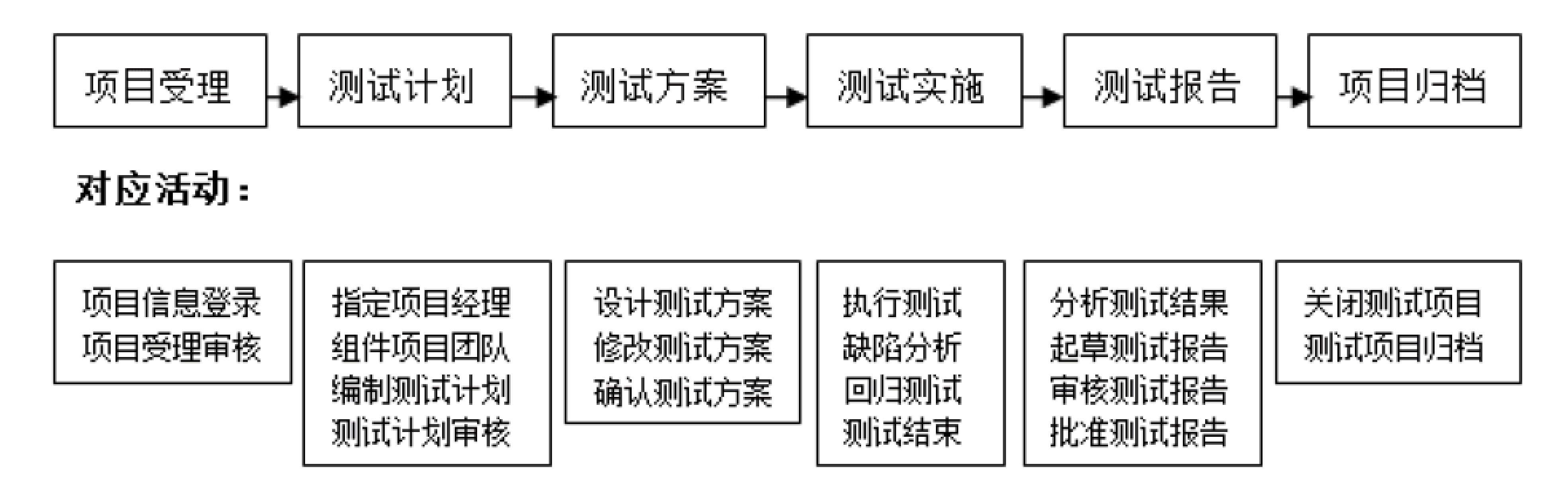


图 1-9　测试活动

根据上述流程，对测试流程的定义固化在平台中，保证了在测试组织范围内一致地执行统一的测试流程，从测试过程上保证软件测试的质量。

该平台体现了测试复用和知识共享的思想。软件测试的复用不仅能提高测试工作的效率，更能解决测试人员经验不足的问题。软件测试的复用主要包括测试过程的复用、测试方法的复用和测试技巧的复用三方面。

测试过程的复用就是测试流程的复用，采用上述既定流程，针对具体被测项目的需求，进行剪裁，使测试流程规范化。

测试方法的复用主要指测试计划的设计、测试策略的采用、测试项细则的编写、测试记录的方式、软件缺陷的分析和测试报告的撰写等方面的复用，而这类复用的具体体现可以采用统一的测试文档模板，可以最大程度避免测试的随意性，提高测试设计的质量。

测试技巧的复用主要指测试用例的复用。在特定情景下，选择何种测试用例是发现软件问题的关键因素，而测试用例的设计与测试人员对被测软件的理解以及经验的积累密切相关。如果将大量的测试用例收集到测试用例库中，经过合理分类，供测试人员选择使用，将极大地提高软件问题的发现率。

测试管理系统常见特点包括：实现对测试需求、测试计划、测试次数、测试集、测试用例、测试缺陷等进行管理，覆盖测试策划、测试设计、测试执行、测试总结等测试过程。

测试管理系统支持根据测试需求对测试工作进行的计划，建立和维护测试需求、测试对象、测试用例、缺陷之间的追踪关系；支持对测试工作进行分析，提取测试的过程数据，生成测试报告；为了适应用户的多变要求以及提高系统的灵活性，系统提供缺陷处理流程定制，以及测试类型、测试级别等的代码定制功能。

4）功能结构

功能结构图如图 1-10 所示。此功能结构的主要特点为：

(1) 满足国际标准关于软件测试的要求。

(2) 支持对需求多层次结构化的管理。

(3) 支持管理测试需求、测试对象、测试用例、缺陷之间的可追溯性，并能够互相进行

追踪。

(4) 支持自定制缺陷处理工作流程。

(5) 支持自定制问题级别、问题类别、测试类型等信息。

(6) 支持基于过程数据，以多种图表方式对测试结果进行分析。

(7) 与其它工具集成。

(8) 提供图形化向导，便于测试人员开展工作。

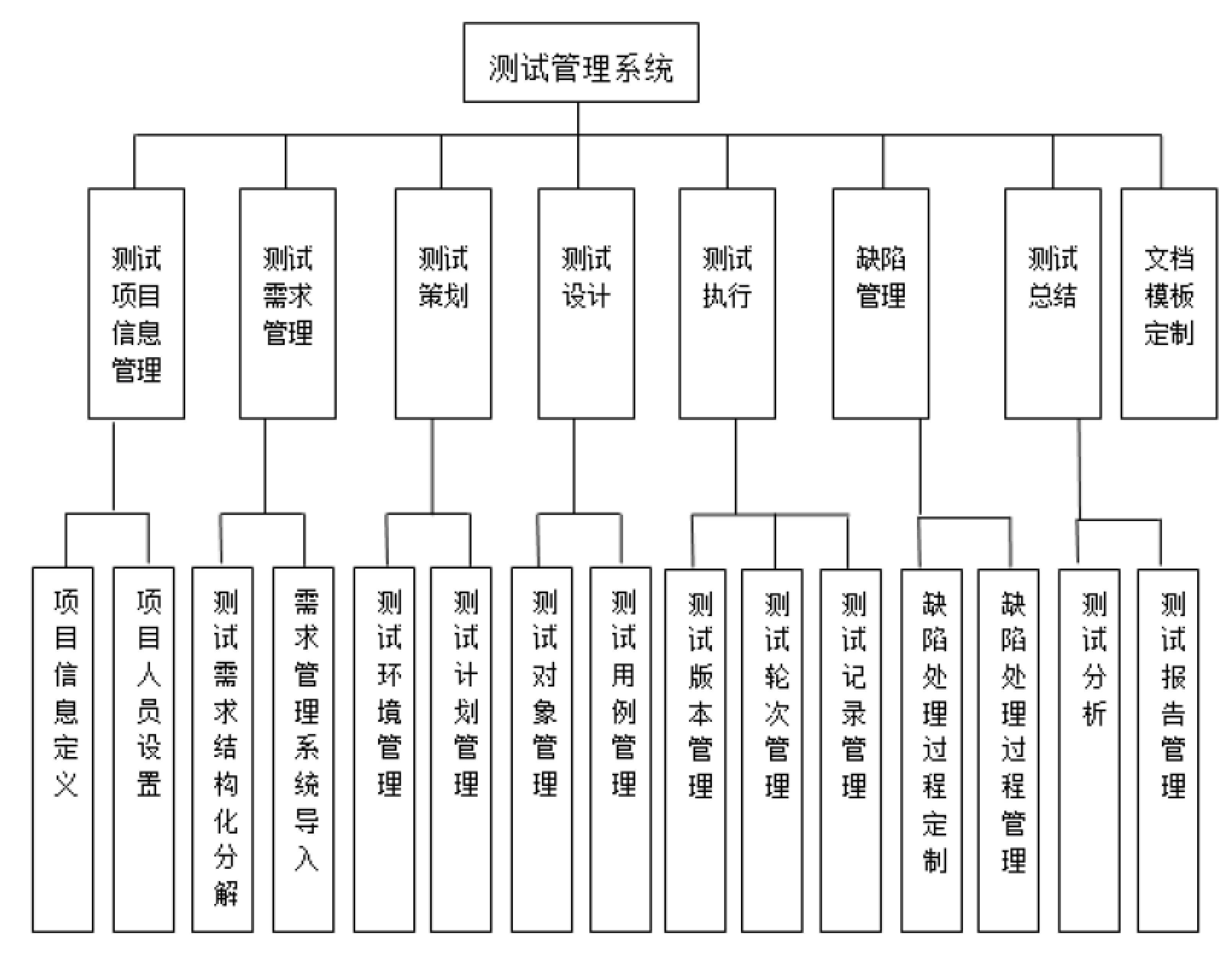


图 1-10　功能结构图

1.3　软件测试管理要素

做好测试管理，重要的是把握住几个关键要素。测试管理的目的主要通过对这些要素进行管理来实现。成功的软件测试项目有很多构成要素，如测试人员的个人能力、管理者的综合素质、公司的开发过程等。测试管理中需要注意所有基本的和风险高的环节。本书对测试文档、测试团队、测试质量、缺陷、测试环境、测试需求、测试流程和测试执行都有专门章节进行具体分析，因此本节重点分析软件测试管理要素定义、要素定位和相互关系。

1.3.1　基本定义

首先，由于软件测试的出发点就是质量，软件测试的一切工作应该围绕质量而开展。质量是软件测试的顶点，如图 1-11 所示。测试的其他部分就是支撑这个顶点的测试人员、测试资源、测试技术和测试流程。因此，构成软件测试的 5 个要素就是：质量、人员、技术、资

源、流程。

1. 质量

质量一词的英文是“Quality”。ISO 8402“质量术语”定义质量为反映实体满足明确或隐含需要的能力的特性总和。软件质量是软件测试的目标，也是软件测试工作的中心，一切从质量出发，也就是一切从客户需求出发。任何违背质量的东西都是问题，测试管理就是要找出这些问题。

图 1-11　测试管理的 5 要素

2. 人员

人员就是测试人员。测试人员因素包括测试组织结构、角色和责任的定义。测试团队管理是对测试过程中涉及的不同角色和人员的管理，包括组织架构、团队建设、测试培训和人员激励措施等。人员即测试团队和成员，是最主要的因素。那么测试团队的管理就成为软件测试管理非常重要的基础。

3. 技术

技术在这里指软件测试技术，包括方法、工具。软件测试工具就是通过一些工具能够使软件的一些简单问题直观显示在读者的面前，这样能使测试人员更好地找出软件错误所在。业界所有的测试执行没有完全手动的，有很多可用的测试工具。本系列丛书主要介绍惠普的软件测试工具。工具不仅仅是软件、服务或平台，还有很多简单常用的业界用来帮助测试工作的工具，比如流程图、模板等。

现在以软件测试管理体系为例说明测试管理要素之一的技术要素。软件测试管理体系本身是软件应用，就是一种技术、一门工具。软件测试管理体系涉及功能测试、性能测试、安全性测试等多种软件测试类型，包含软件测试流程中各阶段的流程规范、测试理论、测试工具等各方面内容。某公司的软件测试管理体系的总体架构如图 1-12 所示。

该测试管理体系是非常实用的测试管理技术，总体上说明了所有重要测试环节和工具。它包含相关规范和相关模板两部分，其中相关规范包含测试管理平台使用的管理规程、软件测试流程规范、软件缺陷管理规范、测试管理平台管理规程、测试相关模板套用规则等。相关规范主要用于规范软件测试流程及测试管理过程，为软件测试过程提供理论支撑。相关模板包含功能测试的测试需求、测试计划、测试用例、测试报告、性能测试方案、性能测试报告，以及测试过程中测试文档评审意见、测试文档修改意见等模板。这些模板为软件的功能测试和性能测试提供统一规范的文档模板，保证软件测试过程中的规范性和统一性。

测试管理平台是组织、管理系统软件测试工作的工具平台，由测试管理工具、自动化测试工具、性能测试工具和辅助工具组成，提供数字化软件测试过程的电子化管理平台，将软件自动化测试、软件性能测试、安全性测试、可靠性测试等测试过程和记录信息集成一体，是非常实用的测试管理技术工具。

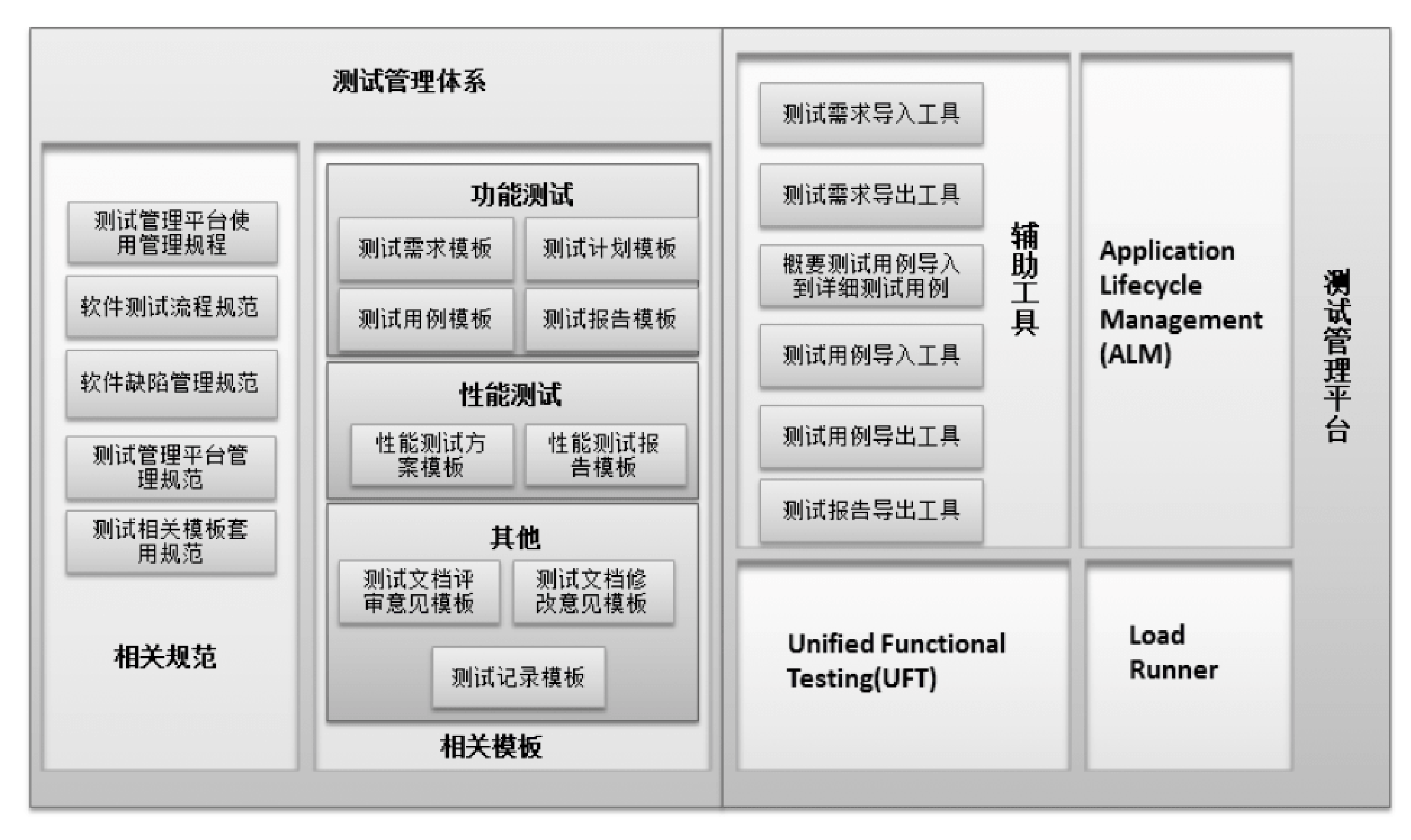


图 1-12　测试管理体系与平台示意图技术及工具必然成为测试管理的要素，因为有适用的技术和工具，才能有效地提高测试的效率、覆盖面以及深度。

4. 资源

软件测试管理中主要考虑资源的可用性和局限性。主要是指测试环境中所需要的预算、硬件设备、网络环境，测试数据、测试时间等可用于投入软件测试的资源。但测试人员在这里没有单独算作资源，虽然人力资源也是资源的一种。但是因为人员的重要性，我们已经把人员列出作为单独的要素，不再包括在资源的要素里。

5. 流程

对测试流程来讲，是从测试计划和测试用例的创建、评审到测试的执行、报告，设定每个阶段的进出等标准或规范，都有关于怎样操作的要求和步骤，这些就是流程。管理测试项目都有个循序渐进的过程，从计划到策略，再到实现。软件流程就是按照这种思维来定义我们的开发过程，根据不同的产品特点和以往的成功经验，定义从需求到最终产品交付的一整套流程。管理测试流程告诉我们该怎么一步一步去实现软件产品的测试，可能会有哪些风险，如何去避免风险。本书有专门的章节讨论测试流程。

流程管理的一个示例就是软件版本控制。版本控制是软件配置管理的初期表现形式，来源于测试对稳定环境的要求。测试版本控制，简单说就是测试版本是否有明确的标识、说明，并且测试版本的交付是在项目管理人员的控制之下有计划完成的。

测试版本的标识用来识别所有的版本。版本号码的用处很多，例如在填写错误报告的时候往往需要提供发现错误的那个版本。在做缺陷分析时，可以利用版本号来区别缺陷和判断缺陷的发展趋势。测试版本交付，测试版本的控制必须纳入测试管理人员的控制之下。常见的形式就是测试管理者控制测试版本的更新和发布。开发人员在看到错误报告之后，总是倾向于马上修正这些错误并且发布给测试工程师做验证。

测试版本的说明是开发人员和测试人员之间交流的有效形式。测试人员可以通过这份文档了解到当前的测试版本中就上一版本而言有哪些显著变化，明确这些之后，测试人员可以更加高效、有针对性地执行测试。

虽然版本控制有很多好处，但从测试管理角度看，过于频繁的版本更新会降低测试的效率。设想你是一名测试工程师，当测试用例刚刚执行到一半的时候突然发布一个新的测试版本，在这样的情况下，已经执行完毕的测试用例是否还需要再次执行一遍呢？为了规避修改代码带来的副作用，我们有必要执行回归测试。质量是有保证了，但效率降低了。测试在进度上被迫延迟了。所以测试版本的控制虽然有助于保证测试版本的统一和测试效率，但不需要过于频繁。要通过流程管理对版本控制进行制约和策划。

1.3.2　相互关系

很多人对埃及的金字塔感兴趣，不仅是因为规模宏伟、结构精密，而且因为金字塔的兴起和演变已成千古之谜。金字塔有许多特征数据，和 13 世纪数学家法布兰斯提到的奇异数字的组合，有许多巧合之处。这些奇异数字的组合是 1、1、2、3、5、8、13、21、34、55、89、144、233…它们中的任何两个连续的比率都接近 0.618，如 3/5、5/8、34/55、55/89、89/144 等。而且金字塔有一个顶点、5 个面、8 条边，总数为 13 个层面，这些特征数据也和上述奇异数字非常吻合。

回到软件测试管理要素，似乎也能借用金字塔的神秘说明其关系。下面就来看一看：5 个要素构成了 5 个面，每个面由 3 个要素构成，代表着软件测试的工作面。

在金字塔构成中，还有每两个要素构成的 8 条边，每条边代表两个要素之间的关系，如何处理这些关系，也就决定着测试能否获得成功。基于要素、工作面、要素之间的关系，我们确定了 13 项软件测试原则、21 个关键域。针对软件测试关键域，每个软件组织可以了解自己在这个领域的水平，持续进行改进。最后，列出目前所使用的各种软件测试方法，并将这些方法应用于软件测试实际工作中。所以软件测试可概括为图 1-13 所示。

1 中心—>5 要素—>5 工作面—>8 关系—>13 原则—>21 关键域—>34 个方法

图 1-13　测试管理金字塔序列图

1. 1 个中心到 5 个要素

质量是软件测试的中心，这毋庸置疑。测试是质量保证的重要手段之一，测试本身就是为质量服务的。测试能否通过，检验标准是用户的需求，也就是质量的标准。所以，在软件测试的 5 个要素中，质量是核心，其他 4 个要素服务于质量，服从于质量。

除了质量，最重要的自然是测试人员。人是决定因素，决定技术和流程的执行。像软件开发这样的智力活动，要强调“以人为本”的管理文化，才能真正发挥每个人的潜力，以最有效的方法完成测试工作。

在软件测试过程中，哪两样东西是我们必须关注的？答案应该是“测试覆盖率”和“效率”。如何保证质量，重要的衡量方法就是测试的覆盖率，包括用户实际需求的测试覆盖率

和软件代码覆盖率。在保证质量的前提下，确定任务的优先级，采取正确的策略和方法，包括自动化测试方法，以高效的方法完成测试。

拿什么来保证“测试覆盖率”和“效率”呢？这不外乎三方面，就是测试人员、测试流程和测试技术。就人员来说，要从招聘、培训和考核等各个环节来培育良好的团队文化，树立正确的工作态度，强化质量意识，提高团队的战斗力，构建卓越的测试团队。无论是采用敏捷的测试流程还是传统的测试流程，一定要结合具体的产品和技术特点，因地制宜，形成适合自己的、有效的测试流程。测试技术比较丰富，因而下面各章的讨论会很多，从客户端到服务器端，从“黑盒”测试到“白盒”测试，从静态测试到动态测试，全力构造完整的测试技术体系，使之满足测试工作的需要。这些内容，可以用图 1-14 形象地描述，使我们一目了然。

1 个中心—>1 人（以人为本）—>2 个目标（关注点）—>3 个支撑—>5 要素或 5 个工作面—>8 关系—13 原则—>21 关键域—>34 个方法

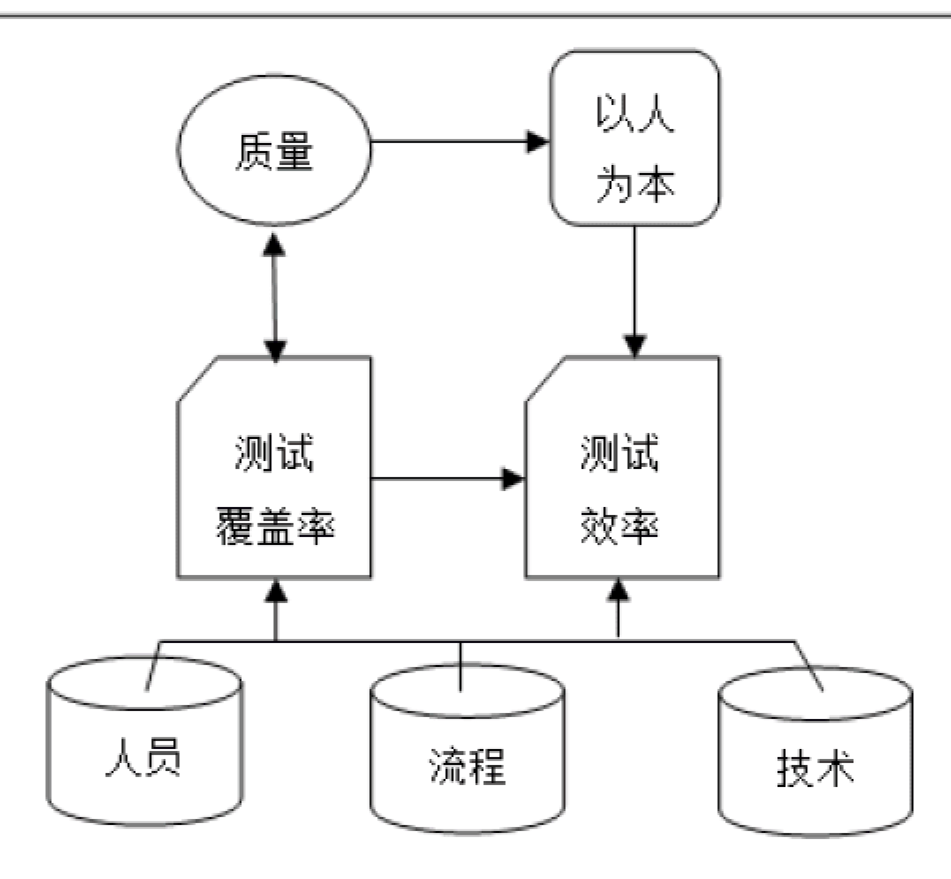


图 1-14　软件测试金字塔和关系实例图

软件测试的金字塔体系可以基于之前的描述进行扩充，得到的结果更接近于一串神秘的数字。

2. 测试管理的 5 个工作面

测试管理的 5 个工作面基于软件测试金字塔的构成，如图 1-15 所示。其中的相互关系分析如下：

1) 质量-人员-技术：团队建设，包括人员的招聘、培训、考核等。

2) 质量-人员-资源：成本管理，人员和软硬件资源都是投入，但同时必须将人和软硬件资源区别对待，不要将人也作为软硬件资源那样处理，否则会带来较多的问题。

3) 质量-技术-流程：技术和流程结合起来就是一种测试架构或测试框架，通过技术，将流程融入系统或工具中，流程的执行才能稳定、有效。技术通过框架固化，技术才能发挥最大效益。

4) 质量-流程-资源：基础设施，构建测试环境，将测试建立在坚固、流程的基础设施

之上。

5) 人员-技术-流程-资源：项目管理需要在一定的质量标准下，如何平衡这些要素以及如何获得最大的生产力，是软件测试项目管理的主要任务。

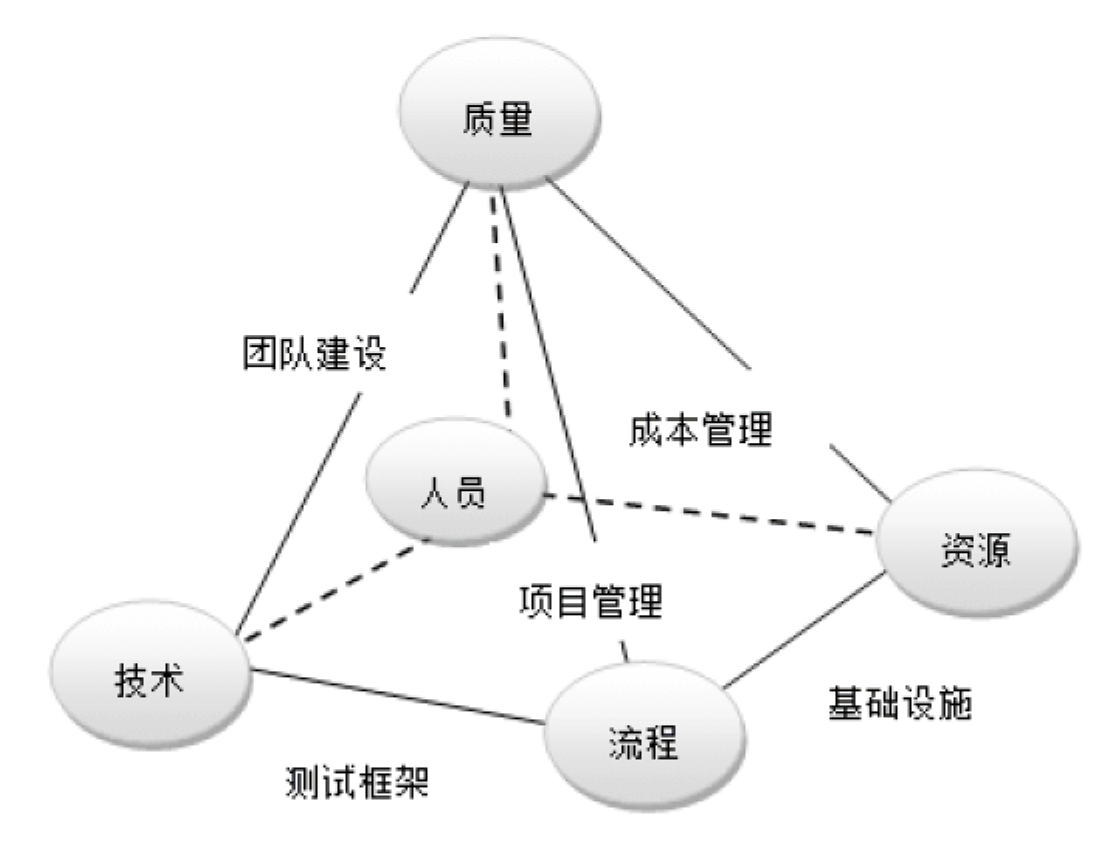


图 1-15　软件测试的 5 个基本工作面

1.4　软件测试管理策略

一种策略必须为使用者提供指南，并且为管理者提供一系列重要的里程碑。软件测试策略就是为测试团队就如何在计划、设计、执行和验收等不同测试环节的框架、流程、范围和注意事项等提出指导性建议。测试策略描述了组织采用的测试方法，包括产品和项目风险管理、测试级别和阶段定义，以及测试的概要活动等。

测试管理策略通常是侧重总体的、高层次的、从始至终的测试工程的总体方法和目标，起指导性的作用。本节将对测试策略做多角度分析，进而说明测试管理策略。

1.4.1　测试管理策略的基本概念与意义

任何完全测试或穷举测试的工作量都是巨大的，在实践上是行不通的，因此任何实际测试都不能保证被测试的软件程序中不遗漏错误或缺陷。为了最大程度减少这种遗漏，同时最大限度发现可能存在的错误，在实施测试管理前必须确定合适的测试方法和测试策略，并以此为依据制订详细的测试计划、测试用例、自动化测试和执行测试等。测试管理策略不是一个人设计出来的，而应该先听取市场、设计、项目经理等相关人员的意见和需求，再结合软件研发和测试需求得出的平衡取舍的产物。

1. 软件测试管理策略的基本概念

测试管理策略是在一定的软件测试标准、测试规范的指导下，依据测试项目的特定环境约束而规定的软件测试管理的原则、方式、方法的集合，需在测试流程、测试计划文档、测试完成标准等信息中体现。

测试管理策略主要回答两个问题：一是测试管理什么，比如测试方法需要包括功能测试(各个功能模块)、性能测试、安全性测试、兼容性测试、文档测试等，即测试范围要明确；二是怎么管理测试，通过哪些管理方法监督和评估功能测试的效果，要考虑基本的理论结合测试需求说明、性能测试的哪些场景、安全测试的级别和内容、不同版本、怎样有效监督和管理。

2. 软件测试管理策略的意义

依据软件管理本身性质、规模和应用场合的不同，我们将选择不同测试管理方案，以最少的软硬件、人力资源投入得到最佳的测试管理效果，这就是软件测试管理策略的意义所在。换句话说，软件测试管理策略就是考虑在现有的资源和条件约束下要完成这个测试项目需要哪些方式和方法，怎样的流程保证，完成的标准是什么，怎样监督和评估，出了问题怎样处理等。

3. 软件测试管理的原则与策略

业界总结的常见软件测试的原则与策略有以下 8 条：

1) 尽早和不断测试。
2) 软件测试应由软件测试专业人员进行。
3) 设计测试用例时应该考虑到合法的输入和不合法的输入以及各种边界条件。
4) 注意测试中的错误集中发生现象。
5) 对测试错误结果有确认过程。
6) 制订严格的测试计划，并把测试时间安排得合理和紧凑。
7) 回归测试时注意关联性，分析有哪些相关受影响的功能，需在回归测试时测试。
8) 进行版本控制，制定变更测试文档的流程。

4. 测试策略的分析过程

测试管理策略的设计是多步骤的分析过程，至少应该包括以下步骤：

1) 测试需求分析
2) 明确测试范围
3) 明确测试资源等制约条件
4) 评估测试风险
5) 确定测试方法和流程
6) 确定测试进入和退出条件

因为测试管理策略的步骤是在软件完成的最终期限的压力已经开始出现的时候才开始进行的，所以测试的进度必须是可测量的，而且问题要尽可能早暴露出来。所以测试管理的策略要体现这方面要怎样测量、监督和管理。

1.4.2　策略对执行软件测试的影响

软件测试管理策略随着软件生命周期的变化、软件测试方法、技术与工具的不同而发生

变化。这就要求我们在制定测试管理策略时，应该综合考虑测试管理策略的影响因素及其依赖关系。软件测试管理策略对软件测试的执行和最终结果都有重要影响。优化的软件测试策略会产生积极的正面影响和结果，当然不实用的测试策略也会导致软件测试执行和最终质量。

当测试策略实用而且得到测试团队的执行和服从时，再加上实用的测试管理体系和工具，理想的影响和结果会包括：

1) 测试管理体系规范从理论和规范上为测试管理过程提供基础，测试管理平台从工具的角度为测试过程管理提供技术支撑，二者融为一体，相辅相成；

2) 测试团队成员服从标准化的测试实施流程，过程快速而高效，真正帮助尽早找到软件缺陷，提升软件的质量；

3) 根据客户的业务需求和软件研发设计需求，对产品或项目的功能、性能、兼容性、易用性、安全性等进行全方位、考虑周到的测试；

4) 测试管理解决方案是一套通用的、应用于多种测试类型的测试管理过程，将自动化测试工具、性能测试工具集成于一体，自动识别所含工具包类型的脚本，融于测试管理工具之中，服务于多种测试类型；

5) 一体化的测试过程管理。测试管理解决方案能够覆盖软件测试过程中的每个环节，将测试产物与测试工具结合起来，不但可以实现省时、省力高效的操作方式，而且更加规范和统一；

6) 基于测试管理体系的测试团队可快速完成软件测试过程，做到和开发团队以及客户需求的紧密结合，共同保证软件产品质量。

习题与思考题

1. 决定软件测试管理目标时需要从哪几方面考虑？
2. 从测试工作流程的角度，列出有哪些主要测试任务？
3. 测试管理包括哪方面的内容？说出对测试成本管理和测试风险管理的理解。
4. 单元测试的定义是什么？单元测试要从哪些方面考虑？
5. 简述集成测试的概念、集成测试常用的类型，并说明每种类型的特点。
6. 什么是敏捷测试？
7. 比较敏捷测试模型和传统的 CMMI 测试的优缺点。
8. 在理解的基础上，分别叙述 TMM 的 5 个等级。
9. 叙述测试管理的关键要素；并说明要素之间的关系。

第2章　软件测试需求管理

需求这个概念并不陌生。软件测试需求描述软件测试工作中“做什么”的问题，是软件测试管理的基础。软件测试工作的进度、风险和资源都基于精确的软件测试需求分析推导得出。软件测试需求分析是软件测试策略、案例、方法、流程、人员等测试任务的基础。

本章将从介绍软件测试需求的概念开始，进而介绍如何分析和管理软件测试需求。

作为应用型软件人才，学习完本章后，应具备以下能力：深刻理解软件测试需求的定义和意义，熟练掌握软件测试需求的分析方法，能在软件测试工作初期，准确地把握软件测试需求，为后续的测试活动打下坚实的基础。灵活运用软件测试需求的管理方法，在实践过程中严格、全面、持续地对测试需求实施管理，根据实际情况进行变更管理，确保软件测试工作始终处在正确的轨道之上。

2.1　软件测试需求概念

在软件开发中，需求是指待开发软件产品的目标用户对该软件产品的功能，性能、设计约束和其他方面的期望和要求。也就是说，软件系统的需求是指用户对软件功能的要求，就是用户希望软件系统能做什么事情，完成什么样的功能，达到什么样的性能等。

如果以一个项目的观点看待软件测试工作，这个项目的范围就是软件测试需求，它定义了软件测试工作的范围，是进行其他测试活动的基础。为了便于用户更好理解，本节将从最广义的需求逐步推演，详细介绍软件测试需求的概念。

2.1.1　软件测试需求

下面首先从软件需求的基本概念入手，再逐步举例说明软件测试需求的概念。

1. 软件需求的定义

定义一：软件需求就是一个为软件系统的需求进行启发、组织、建档的系统方法，一个建立和维护客户和项目团队之间关于变更系统需求所达成的一致性的过程。他把需求定义为“(正在构建的)系统必须符合的条件或具备的功能”。

定义二：一种获取，组织并记录系统需求的系统化方案，以及一个使客户与项目团队对不断变更的系统需求达成并保持一致的过程。

著名的需求工程设计师 Merlin Dorfman 和 Richard H. Thayer 提出了一个包容且更为精练的定义，它特指软件方面——但不仅仅限于软件：

"软件需求可定义为：用户解决某一问题或达到某一目标所需的软件功能。系统或系统构件为了满足合同、规约、标准或其他正式实行的文档而必须满足或具备的软件功能。"

测试需求并不等同于简单的测试范围，也不是测试设计。因此也有专家定义测试需求不是对测试提出的要求的总和，而是根据程序文件和质量目标对软件测试活动所提的要求。不同的软件有不同的测试需求，不同的用户有不同的测试需求，不同的公司有不同的测试需求。

测试需求不同于测试设计。按照 IEEE 标准，测试设计的目的是：细化测试计划中描述的测试途径，确定要包含的特性和测试，确定完成测试所用到的测试用例和测试规程，最后给出测试失败和通过的标准。

测试需求也不同于测试范围。因为测试范围实际上是在明确了测试需求和测试目的之后要决定的到底测试些什么，不测试些什么。测试需求一般可能比较笼统，而测试范围很具体。比如，某外包公司承担了一个测试项目。测试需求是要在 10 天内完成对一个公司网站的测试。又给出有限的预算和大概的质量要求。比如所有的网页必须在主要浏览器上显示正常。而测试范围就是哪些浏览器要测试到、哪些测试用例要执行等。所以测试需求要视组织的质量目标而定。

2. 软件测试需求的内容

软件测试需求管理恰如裁缝量体裁衣，直接关系到最终测试结果的成型。仅从字面出发，如果一个产品满足客户需求，那它无疑就是成功的。通过对测试需求管理在测试项目进程中实施的不同测试任务进行分析，以便能够确认我们明确客户或开发项目负责人对测试的需求是什么(质量)；满足客户需求的最佳解决办法(统一性)。

明确软件测试需求就能帮助确定测试工作的范围，这是策划和编写测试用例的依据，测试需求分解得越详细精准，表明对所测软件的了解越深，对所要进行的任务内容就越清晰，对测试用例的设计质量的帮助越大。详细的测试需求还是衡量测试覆盖率的重要指标，测试需求是计算测试覆盖率的分母，没有详细的测试需求就无法有效进行测试覆盖率的计算。

3. 实例说明

软件测试需求的内容包括功能、性能、可靠性、国际化与本地化，以及安全等多方面。下面举例说明。

1) 功能测试需求

功能测试需求描述软件测试工程师和其他相关人员对功能性测试的要求、约束等。以网上商城购物车的商品数量这个功能为例，如果当前商城内该商品可销售数量为 100，熟练掌握等价类划分和边界值分析的测试工程师明白，负数、0、小于 100 的正整数、100、大于 100 的正整数以及超出正整数上限的数，这些就是功能测试需求。另以软件测试中的数据边界值为例。要对数值 0 进行测试；还要对与 0 非常接近的负数进行测试，这就是两个具体的测试需求。

2) 性能测试需求

以网上商城商品搜索功能为例，当服务器端收到搜索条件后，后台服务器应在两秒内向客户端返回查找数据，不考虑服务器端到客户浏览器端的性能损失。

3) 可靠性测试需求

例如，网上商城服务器出现不能响应来自浏览器查询请求的平均失效间隔时间 MTBF 应大于等于 3 个月。

4) 国际化与本地化测试需求

例如网上商城应支持中文和英文，不过这样的需求过于笼统，需要进一步细化，比如检查中文版网上商城网页界面上的文字是否有截断情况。

5) 安全性测试需求

例如，网上商城用户密码强制为至少 7 位且包含大写字母、小写字母、数字和特殊字符的强密码，缺少任意条件，单击注册按钮后应提示用户按密码格式要求输入新密码。

除了软件测试需求的内容，从如何确定软件测试需求的角度，有定义、评审等过程。从管理的角度还包括软件测试需求的存储与分类、状态及状态转换、软件测试需求的跟踪以及变更等。这些内容会在后面详细介绍。

2.1.2　软件需求管理

软件的需求管理(Requirements Management)是软件开发过程中最难把握的一个环节，也是项目成败的关键因素。在整个软件生命周期中，需求阶段是基础。当今，尽管有关开发的知识和经验不断丰富，可利用的工具也不断增多，但仍然有相当比例的软件开发项目失败，原因常常是在开始时没有正确地确定和定义软件需求，或者是随着项目的展开没有正确地管理软件需求。

做好需求管理，既可以减少软件开发中的错误，保证项目能满足用户需求，还可以减少修改错误的费用，从而大大缩短软件开发时间，提高软件开发效率，降低软件开发成本。

怎样才能合理并且有序地把我们的软件需求管理起来呢？要建立一套完整的软件测试需求管理机制，对需求从产生到消亡整个生命过程进行监督和管理，合理有效地分配公司有限的人力、物资资源，才能有效地减少由于软件测试需求不明晰而造成的资源浪费和项目失败。

图 2-1 体现了一切从需求开始的具体软件需求管理的流程和步骤说明：

软件需求管理通常需要达到以下目标：

(1) 软件需求规格说明已文档化，并经评审后存档。

(2) 文档化的软件需求规格说明受管理和控制。“受管理和控制”就是在给定时间使用的软件产品的版本均是可查知并受控的(版本控制)，而且以受控的方式进行改动。如果需要比此更高程度的控制，则产品可置于软件配置管理的严格控制之下，如在 CMM2 级的软件配置管理关键过程域中所述。

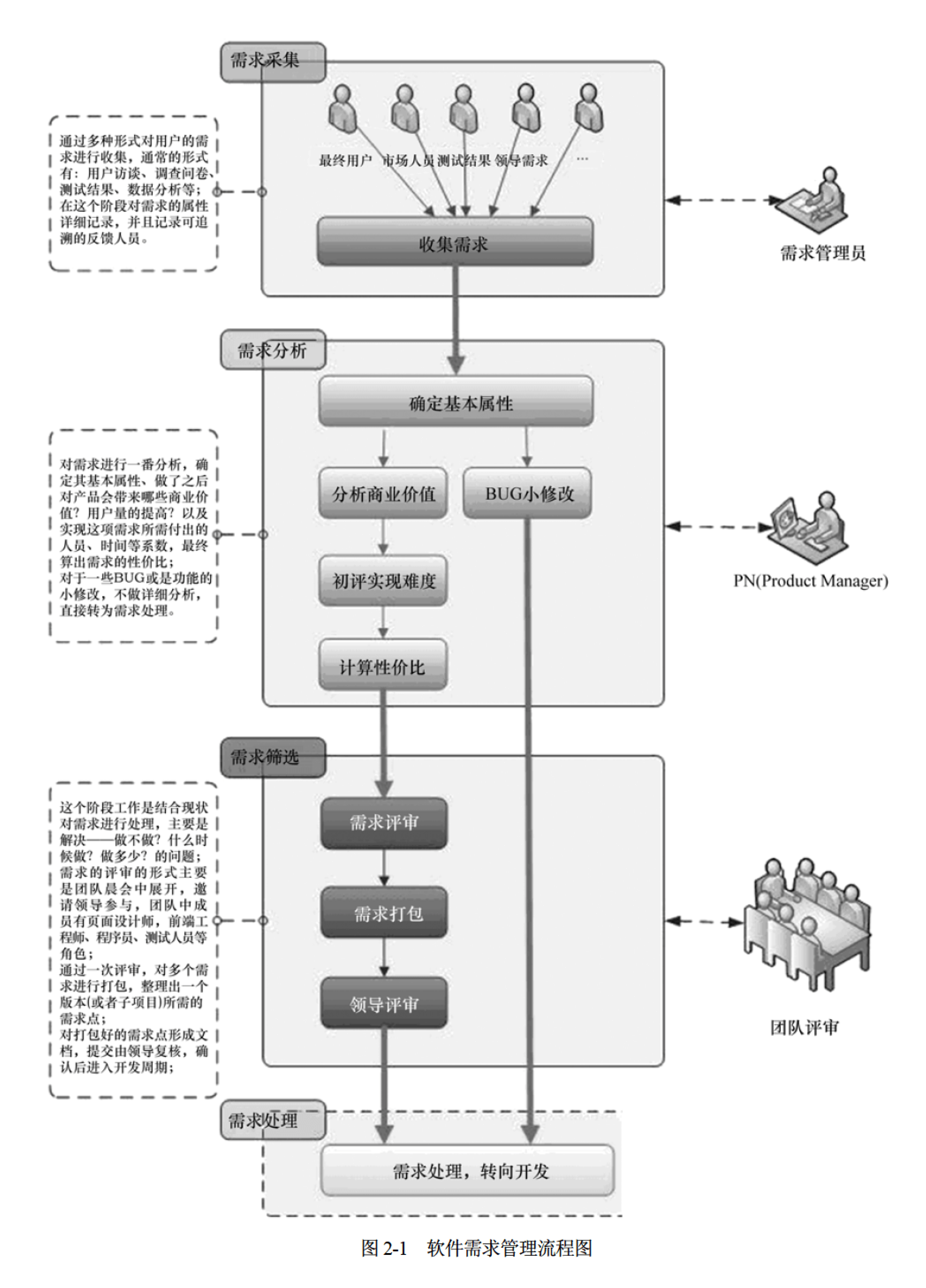


图 2-1　软件需求管理流程图

(3) 供软件工程和管理使用的分配基线已建立，使软件产品满足分配需求的接收标准；分配需求是制订软件开发计划的根据，是整个软件生命周期中估算、计划、执行和跟踪软件

项目活动的基础。

(4) 软件开发计划、软件工作产品和软件过程活动与软件需求保持一致。

软件需求管理就是对分配需求进行管理，主要体现在以下几方面：

(1) 在客户和实现客户需求的软件项目之间达成共识。

(2) 控制系统软件需求，为软件工程和管理建立基线。

(3) 保持软件计划、产品和活动与系统软件的一致性。

归纳起来，软件需求管理就是要确定便于所有整个研发、项目管理和测试团队理解的需求，稳定软件需求并说明需求的更改对项目成本和日程的影响。

2.1.3 软件测试需求管理

前面的内容已经说明了什么是软件测试需求和软件需求管理，本节讨论软件测试需求管理。为提高客户满意度需要提高产品质量，作为质量保证的重要一环，测试人员需要站在客户立场做测试，那就需要首先明确应该测试什么的问题。测试需求分析的目的是明确测试的范围。

软件测试需求管理是要通过人为的和技术的手段、方法和流程，以保证和监督测试团队明确测试软件产品的目标。也就是说，测试工程师以及开发工程师、项目经理、高层管理团队等关心测试产出物的相关人员对测试该软件产品的功能，性能，安全性、可靠性、设计约束和其它等方面的期望和要求。

测试需求管理本就是一个动态的过程，业界常见的测试过程中存在以下问题：

(1) 软件测试本身的需求文档不全，有些需求待定；

(2) 产品质量维度事先没有全面明确，要包括的测试类型不全；

(3) 测试计划和编写测试用例没有规范；

(4) 没有系统的测试需求分析方法、流程或指导；

(5) 测试中经常会出现需求和缺陷遗漏、测试设计遗漏的问题。

软件测试需求管理就是要应对软件需求、软件测试需求及相关需求的问题，并有效地分析出测试的具体需求，并以此为软件测试设计提供尽可能准确的信息作参考。

2.1.4 软件测试需求管理的意义

软件测试需求管理(Software Testing Requirement Management)是整个软件测试管理体系中重要的一环。一套软件测试需求管理应当是待测试软件产品需求的完整体现，每部分测试任务都是对总体需求一定比例的满足(甚至是充分满足)，仅仅解决部分需求是没有意义的。对关键测试需求的疏忽很可能是灾难性的。本节从一些业界研究报告结果出发，分析目前软件开发项目中存在的问题，从以下几方面说明软件测试需求管理的必要性和意义。

1. 失败经验教训说明必须要有软件测试需求管理

消除软件开发百病之源的有效措施是管理软件测试需求，总结我们之前做过的产品，特别是不成功的产品，总是能找到其中的一些原因。而在这些原因中我们不难发现对于因没有

合理管理好测试需求而导致的问题不在少数，所以就会出现类似于“测试人员并不清楚究竟该测什么，但却一直忙碌不停地测试”的情况，到最终通过测试的产品还是达不到用户要求或者不符合所开发软件的设计需求。随着计算机和信息技术的发展，软件测试工作越来越复杂，规模越来越大，而且随着软件企业的日益壮大，软件测试需求管理已成为日常软件开发中不可或缺的组成部分。

在软件项目的测试过程中，测试需求管理贯穿软件项目的整个生命周期，是软件项目管理中一项十分重要的工作，据调查显示在许多失败的软件项目中，由于测试需求不清而导致的约占四至六成，因此，测试需求管理是提高软件质量的基础，甚至影响到软件项目的成功与否。

经验证明，测试需求是测试工作的根源，测试需求的优劣对产品的影响最大；它规定了测试工作的范围如果一开始方向错了，接下去的工作就会错上加错，离“好产品”越走越远，所以我们先抓源头，再有序地开展产品测试过程。

2. 业界专家建议必须实施软件测试需求管理

简单地说，软件研发团队之所以管理需求，是因为他们想让项目获得成功。满足项目需求即为成功打下基础。若无法管理需求，达到目标的几率就会降低。以下最近收集的证据很有说服力：Standish Group 从 1994 年到 2001 年的 CHAOS Reports 证实，导致项目失败的最重要原因与需求有关。2001 年，Standish Group 的 CHAOS Reports 报道了该公司的一项研究，该公司对多个项目作调查后发现，74%的项目是失败的，即这些项目不能按时按预算完成。其中提到最多的导致项目失败的原因就是“变更用户需求”。

需求分析作为软件生命周期的第一个阶段，并贯穿于整个软件生命周期，其重要性越来越突出。到 20 世纪 80 年代中期，逐步形成软件工程的子领域——需求工程。进入 20 世纪 90 年代后，需求工程成为软件界研究的重点之一。从 1993 年起，每两年举办一次需求工程国际研讨会(ISRE)，1994 年起，每两年举办一次需求工程国际会议(ICRE)。一些关于需求工程的工作小组也相继成立。

在软件项目管理中，需求工程是软件开发的第一步，是关键的一步，也是最难把握的一步。需求管理做得好坏直接影响到软件的质量，甚至软件项目的成败。从软件的项目立项、研发到维护，用户的经验在增加，对使用软件的感受有变化，以及整个行业的新动态，都为软件带来不断完善功能、优化性能、提高用户友好性的要求。在项目管理过程中，项目经理经常面对用户的需求变更，如果不能有效处理这些需求变更，项目计划会一再调整，软件交付日期一再拖延，项目研发人员的士气将越来越低落，将直接导致项目成本增加、质量下降及项目交付日期推后。这决定了项目组必须拥有需求管理策略。相应的，测试需求管理也要跟上。图 2-2 说明一个简单的测试流程有 6 个主要环节。在整个软件测试流程中，分析测试需求是第一步。

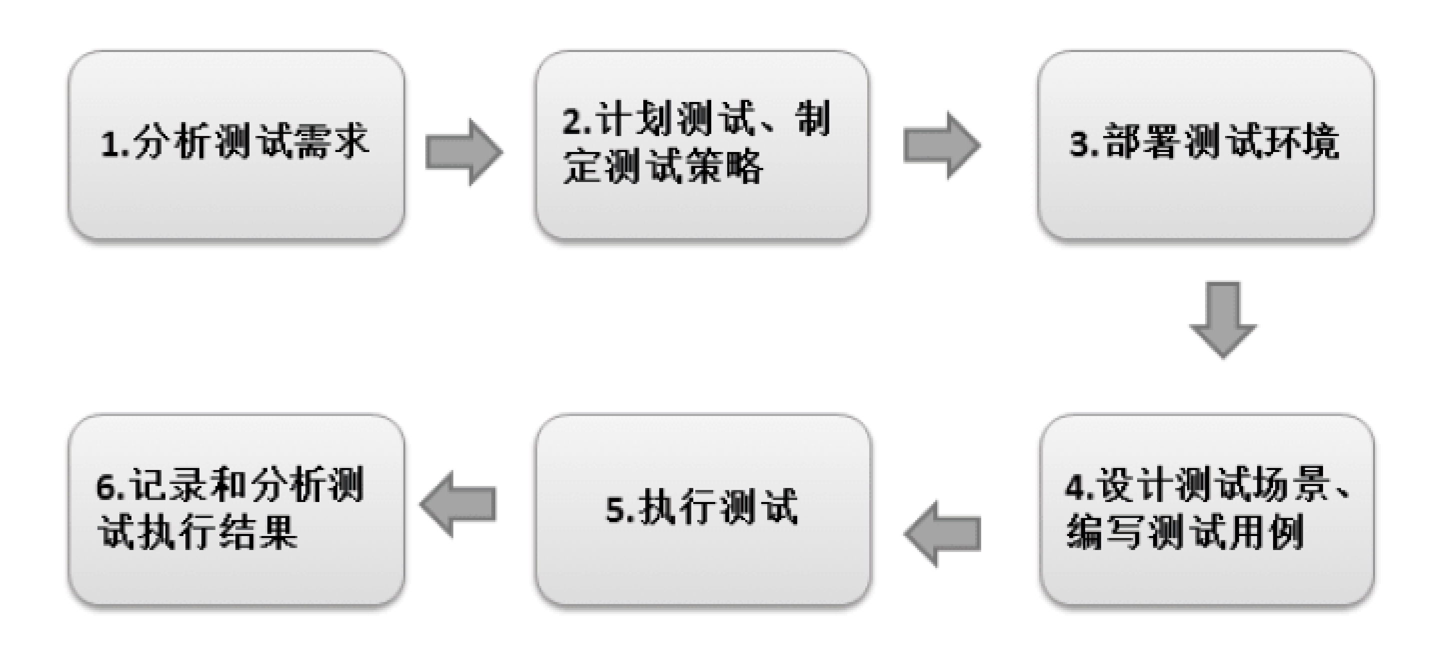


图 2-2　软件测试的 6 个主要环节

3. 缺乏成熟的理论和统一标准而需要需求管理

目前，软件工程仍然是一个新兴的工程领域，远远没有其他传统工程领域那么规范，其开发过程缺乏成熟的理论和统一的标准。很多软件测试活动中都存在以下列出的问题：

(1) 忽视开展软件测试活动前的测试需求分析；

(2) 测试过程缺乏统一、规范化的方法论的指导；

(3) 测试文档资料不齐全或不准确；

(4) 忽视与用户之间、开发组成员之间的交流；

(5) 忽视测试的重要性；

(6) 不重视维护或由于上述原因造成维护工作的困难。

2.2　软件测试需求分析

如何获取测试需求？如何策划软件测试需求？如何对测试需求进行评审？如何控制需求变更？如何利用软件工具的辅助来支持测试需求管理？这些都需要进行测试需求分析。测试需求分析是软件测试过程中一项极为重要同时又极为复杂的重要环节。做软件测试需求分析也是很复杂和有挑战性的。通常有难以确定和易变的特点，而软件测试需求的变化也会直接影响软件测试结果和效果。本节侧重怎样具体进行软件测试需求分析的方法、需求分析的过程、结果与评审。

2.2.1　分析的目标和任务

如果把软件研发过程中的测试活动看成单独的项目，软件测试需求分析就是这个项目正式启动后的第一个环节，是项目成败的关键因素。在整个软件测试生命周期中，测试需求阶段是基础。

软件测试需求分析的目的就是对软件测试要解决的问题进行详细分析，弄清楚参与软件测试活动的干系人对软件测试活动和交付物的要求，包括需要输入什么数据，要得到什么结果，最后应输出什么。

软件测试需求分析有助于确定软件测试活动的范围。只有这样才可以减少软件测试中的

错误，保证软件测试活动能满足与之相关的干系人的需求，还可以减少甚至避免由于测试工作初始方向性错误引入的返工，节约软件测试开销，缩短软件测试时间，提高软件测试效率，降低软件测试成本。

软件测试活动的干系人组成非常复杂。做好测试需求分析的意义很重要，最直接的参与者是软件测试团队成员、开发团队和项目经理。再向外扩张，软件的最终用户和公司的高层管理团队都和软件测试活动息息相关。用户可能不熟悉计算机的相关知识，而软件开发人员对相关的业务领域也不一定有深刻的理解，用户与开发人员之间对同一问题理解的差异和习惯用语的不同往往会造成产品的功能、性能等无法满足用户真正的要求。测试需求分析就是架起用户和软件产品之间的桥梁，保证产品在交付客户的时候达到质量标准，提高用户满意度。

测试需求阶段要解决的问题是让软件测试团队和测试活动相关干系人共同明确将要进行的测试活动的要求。具体而言，测试需求分析主要有两个任务：第一个是通过对测试活动需要解决问题及环境的理解、分析和综合，建立分析模型；第二个是在完全弄清所有测试活动干系人对测试的确切要求的基础上，用“软件测试需求规格说明书”(Software Test Requirement Specification，SRS)把测试需求以正式书面形式确定下来。以下讨论几个重要的软件测试需求分析环节：

1. 建立软件测试需求模型

测试需求分析的一项任务是建立软件测试需求模型。初步接触软件测试工作的人都对测试活动表面上杂乱无章有很深刻的印象，但是受过严格训练、有经验的软件测试工程师总可以通过分析与归纳从中找出一些规律，再通过“抽象”建立起系统的模型。

软件测试需求分析模型是描述软件测试需求的一组模型。由于测试活动干系人往往会从不同的角度阐述他们对此测试活动或结果的要求，因此有必要为软件测试活动建立模型。这种模型一方面用于精确地记录测试活动干系人对原始问题和目标软件的描述；另一方面，也将帮助分析人员发现用户需求中的不一致性，排除不合理的部分，挖掘潜在的干系人需求。这种模型把来自干系人对测试需求的非技术性描述转为具体、明确、使用技术语言描述的测试需求。这种模型往往包含问题及环境所涉及的信息流、处理功能、用户界面、行为模型及设计约束等，是形成需求说明、进行软件测试活动计划、设计与实施的基础。

2. 编写测试需求说明书

测试需求分析的另一项任务是编写测试需求书。测试需求说明书应该具有准确性和一致性。因为它是连接计划时期和开发时期的桥梁，也是设计测试用例的依据。如果测试需求说明书存在遗漏、含糊不清、前后矛盾，轻则导致测试工程师在开发测试用例时需要付出额外的代价和沟通成本，重则可能导致漏测、误测，导致严重的缺陷在软件发布后影响用户使用，造成实质性损失。

测试需求说明书应该具有清晰性和无二义性。因为是沟通用户和测试工程师思想的媒介，双方用它来表达对于需要计算机解决的问题的共同理解。如果在测试需求说明书中使用了用户

和实际实施测试工作的测试工程师不易理解的专门术语，或用户与分析人员对要求的内容可以做出不同的解释，便可能导致系统失败。测试需求说明书应该直观、易读和易于修改。为此应尽量采用标准的图形、表格和简单的符号来表示，使不熟悉计算机技术的用户也能一目了然。

2.2.2　分析的方法

如果想成功做一个测试项目，首先必须了解测试活动的内容、规模、复杂程度与可能存在的风险，这些都需要通过详细的测试需求分析来了解。所谓知己知彼，百战不殆。测试需求不明确，只会造成获取的信息不正确，无法对被测软件有清晰全面的认识，测试计划就毫无根据可言，进而执行测试也就成为空中楼阁。

测试需求越详细精准，表明对所测软件的了解越深，对所要进行的任务内容就越清晰，就更有把握保证测试的质量与进度。如果把测试活动比作软件生命周期，测试需求就相当于软件的需求规格，测试用例相当于软件的详细设计，测试用例的开发过程相当于软件的编码过程。但是软件测试不是简单把“软件”两个字全部替换成“测试”这样简单，就测试需求分析的方法而言，既可以参考软件需求分析方法，又有其独特性。以下以实例说明具体方法。

1. 从软件需求推导软件测试需求的方法

通过软件需求推导软件测试需求是软件测试需求分析最通用的方法。相比于单纯依赖于测试设计人员的测试经验，由此方法得出的测试需求、测试用例设计比较充分，测试的目的性强，软件需求测试覆盖度高，不容易产生遗漏。

为实现上述目的，本方法提供了软件测试需求分析的具体步骤，如图 2-3 所示。

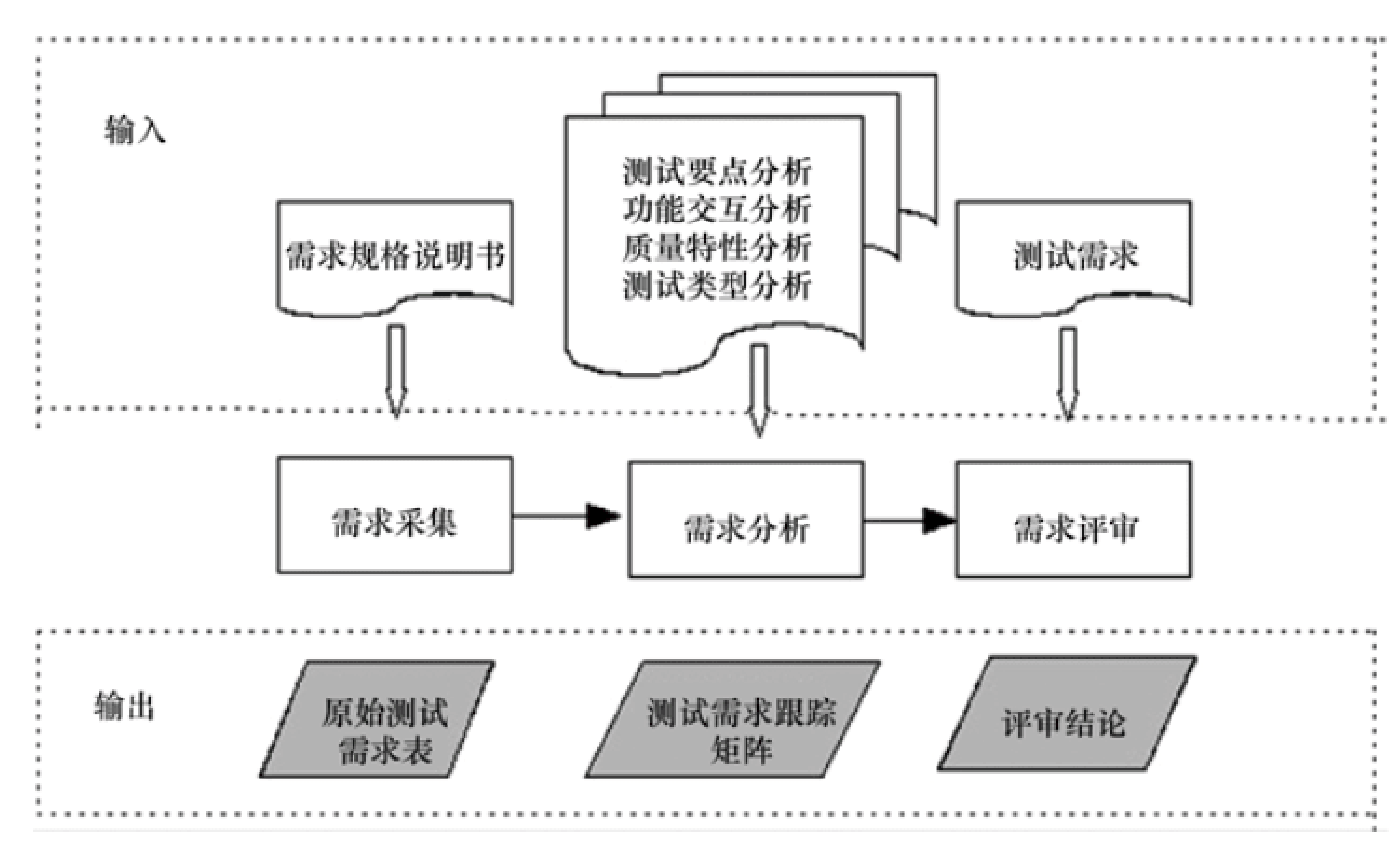


图 2-3　软件测试需求分析步骤

(1) 根据软件开发需求说明书逐条列出软件开发需求，并判断其可测试性。如果不具备可测试性，则需要提交申请对软件开发需求说明书进行变更，任何软件开发需求都应具备可测试性(Testability)。通常来说，对软件开发需求说明书的可测试性检查应该在软件开发需求说明书的评审过程中提出并解决。

(2) 对步骤(1)中列出的每一条开发需求，形成可测试的描述。针对这条开发需求需要进行测试范围的界定。开发需求和需要进行测试的范围不是 1∶1 的关系，可能是 1∶N 或 N∶1，必要情况下，需要对开发需求进行分解和合并。

(3) 对步骤(2)中形成的每一条测试范围，根据质量标准，逐条制定质量需求，即测试通过标准，用以判断测试成功和失败。

(4) 对步骤(3)确定的质量需求，分析测试执行时需要实施的测试类型，至此形成专业的测试需求。

(5) 建立测试需求跟踪矩阵，输入测试需求管理系统，对测试需求实施严格有效的管理。

2. 应用实例说明

下面以一个实例加以详细说明。

建立开发需求列表。表 2-1 是对网上商城购物车的商品数量功能开发需求的描述。将每一条软件需求对应的开发文档及章节号作为软件需求标识，使用软件需求的简述作为原始测试需求描述，没有文档来源的开发需求可用隐含需求或遗漏需求进行标识，标明软件需求获取的来源信息，如开发文档、相关标准、与用户或开发人员的交流等。

表 2-1　网上商城购物车的商品数量功能开发需求描述

需求 ID	需 求 标 识	需求详细描述
256	网上商城购物车的商品数量功能	网上商城购物车的商品数量功能为用户提供购买多件同一商品的功能。商品数量的初始值为“1”，用户可以通过单击“-”、“+”按钮分别减少或增加一件商品数量，也可以通过键盘直接输入数字，比如直接输入“100”

由于在提取的开发需求中可能存在重复和冗余，可通过以下方法整理开发需求：

- 删除：删除原开发需求列表中重复的、冗余的含有包含关系的开发需求描述；
- 细化：对太简略的开发需求描述进行细化；
- 合并：如果有类似的开发需求，在整理时需要对其进行合并。

在表 2-1 中，对于每一条开发需求，从测试角度来考虑，形成可测试的分层描述的测试需求。具体地，通过分析每条开发需求描述中的输入、输出、处理、限制、约束等，给出对应的验证内容；通过分析各个功能模块之间的业务顺序和各个功能模块之间传递的信息与数据，对存在功能交互的功能项，给出对应的验证。

对每一条测试需求，从软件质量角度出发，确定所对应的质量子特性。也即从适合性、准确性、互操作性、保密安全性、成熟性、容错性、易恢复性、易理解性、易学性、易操作性、吸引性、时间特性、资源利用性、易分析性、易改变性、稳定性、易测试性、适应性、易安装性、共存性、易替换性和依从性方面的定义出发，确定每一条测试需求所对应的质量子特性，如表 2-2 所示。

软件测试可以划分为以下测试类型：功能测试、安全性测试、接口测试、容量测试、完整性测试、结构测试、用户界面测试、负载测试、压力测试、疲劳强度测试、恢复性测试、配置测试、兼容性测试、安装测试等。

表 2-2　质量子特性与测试类型的关系表

质量特性分类	质量子特性分类测试内容	对 应 关 系	传统分类测试内容
功能性	适合性方面		功能测试
	准确性方面		
	互操作性方面		安全性测试
	安全保密性方面		
	功能性依从方面		接口测试
可靠性	成熟性方面		
	容错性方面		容量测试
	易恢复性方面		
	可靠性依从方面		完整性测试
易用性	易理解性方面		
	易学性方面		结构测试
	可操作性方面		
	吸引性方面		用户界面测试
	易用性依从方面		
效率	时间特性方面		负载测试
	资源利用方面		
	效率依从性方面		
维护性	可分析性方面		压力测试
	易改变性方面		
	稳定性方面		
	易测试性方面		疲劳强度测试
	维护性依从方面		
可移植性	适应性方面		恢复性测试
	易安装性方面		配置测试
	共存性方面		
	易替换性方面		安装性测试
	可移植性依从方面		兼容性测试

不同的质量子特性可以确定出不同的测试内容，这些测试内容可以通过不同的测试类型来实施。例如，从易安装性方面考虑，测试内容包括测试软件安装的工作量、安装的可定制性、安装设计的完备性、安装操作的简易性、是否容易重新安装，这对应测试类型中的安装测试，通过安装测试可以验证这些测试内容。

本方法的一个实施案例是建立质量子特性与测试类型的关系表，参见表 2-2，该表给出了质量子特性与测试类型的对应关系。对所确定的质量子特性，可以使用该表来确定测试类型。

建立测试需求跟踪矩阵，参见表 2-3，将上述步骤分析、确定的开发需求、测试需求、测试类型填入测试跟踪需求矩阵。

表 2-3　测试需求跟踪矩阵

软件开发需求			软件测试需求		
开发需求 ID	开发需求标识	开发需求详细描述	测试需求 ID	测试需求	测试类型
256	网上商城购物车的商品数量功能	网上商城购物车的商品数量功能为用户提供购买多件同一商品的功能。商品的数量初始值为“1”，用户可以通过单击“-”、“+”按钮分别减少或增加一件商品，也可以通过键盘直接输入数字，比如直接输入“100”	256001	检查当购物车商品数量为合法值时，能够提交有正确商品数量的订单	功能测试
			256002	检查当购物车商品数量为非数字时，能够提示用户输入数字型商品数量	功能测试
			256003	检查当购物车商品数量为非合法数字时，如 0、负数等，能够提示用户输入合法的商品数量	功能测试
			256004	检查当购物车商品数量超过剩余商品数量时，能够提示用户输入合法的商品数量	功能测试
			256005	验证单击“-”或“+”按钮时，商品数量可以减少或增加一件	功能测试
			256006	典型用户环境(硬件和软件)验证单击“-”或“+”按钮时，商品数量可以减少或增加一件，界面响应时间应不超过 0.5 秒	性能测试
			256007	验证购物车功能在 IE、Firefox 和 Chrome 三种浏览器下都可以显示正常，并且功能正常	兼容性测试

但是单纯从软件开发需求进行软件测试需求分析可能会漏掉某些隐含的或由于计算机技术方面引入的限制等。比如，存储购物车中商品数量的后台数据类型是否满足商品数量，

如果后台数据类型是 8 位整数，即使是无符号整数，最多也只能处理 256 个商品。在进行测试需求分析的时候也要考虑类似的情景。以下侧重分析性能测试的需求。

性能测试的需求分析有自身的特殊性，以下将以实例说明。一般来说，在软件开发需求说明书中都会给出明确的性能指标，比如单位时间内访问量要达到多少、业务响应时间不超过多少、业务成功率不低于多少、硬件资源耗用要在合理的范围内，这些指标都会以可量化的数据进行说明以达到具备可测试性(Testability)。

如果软件开发需求说明书没有规定性能相关需求，测试经理需要与项目经理和开发经理共同商讨，首先制定系统性能指标。在实际工作中，常见的情况是项目经理和开发经理由于某种原因没有定义性能需求或没能发掘潜在的性能需求。测试经理应配合项目经理和开发经理制定性能需求。

性能测试需求分析一般要经过分析提取指标、建立业务模型和评审指标三个过程。

一般来说，性能测试指标的分析与提取的重点在于最常用、最重要的功能，以及与外部系统交互较多的功能，以大型网上商城为例：

用户登录就是最常用的功能。由于 Cookie 的存在，虽然不一定每次用户都会手动登录，但是如果在网络情况良好的情况下，由于系统原因登录消耗时间达到分钟级，而竞争对手的登录时间只有几秒钟，再忠诚的用户也会失去耐心。所以用户登录要作为性能测试的重点。类似的功能还有单击某一商品后的显示速度和商品查询等功能。

完成商品交易则是最重要的功能，只有不断完成商品销售，网上商城才能实现盈利。完成交易看似简单，后台的业务逻辑异常复杂，即使经过简化以后也要面对一系列数据库的问题。当用户下单后，后台系统首先要查询后台数据库里该商品的剩余库存量是否足够，只有足够，才能向用户反馈可以完成订单。但是这一过程可能面临多个用户同时购买同一商品，造成总购买量大于库存数量的情况，这就需要对数据库加锁，加锁规则过严会导致虚假售罄，甚至成为黑客攻击的薄弱口，加锁规则过松，又可能造成库存不足仍然生成有效订单。不同的网上商城系统的策略均不同，对性能的影响也不尽相同。如果背后逻辑过于复杂，有可能造成性能下降，破坏用户体验。

支付则是与外部系统交互最多的功能。完成支付流程的性能，不仅仅取决于系统本身的算法逻辑是否合理高效、系统是否处于良好状态，也取决于调用银行网上支付系统的性能。此处性能测试的意义不仅仅在于提高用户体验，更是在遇到性能低下的时候，分析造成性能下降的根本原因的有效方法。如果是系统本身的问题，可以迅速修复，如果是银行网上支付系统造成的性能低下，则需要依靠性能测试报告与银行相关部门进行交涉。

除了在技术层面进行分析，还需要针对用户对系统的使用做扎实统计分析。一般大型网上商城，在中午午休和晚上 7 点到 11 点左右这些时间段访问量比较高，而凌晨期间访问量和交易量都大幅下降。测试工程师应对用户的访问规律进行统计和预测，合理推荐或制定性能测试指标，在一定的成本和开发周期影响下，性能测试指标不一定越高越好。

下面就给出一个典型的性能需求示例。

表 2-4 给出的指标描述清晰，通过标准明确，在测试过程中，我们只需收集用户登录模块的响应时间、登录成功率、并发数、CPU 使用率、内存使用率的数据，然后与表 2-4 中的

指标进行比较即可，通过的，就认为达到了客户要求的性能，未达到就分析原因，并给出测试报告及解决建议。

表 2-4　性能需求示例

功能	响应时间	业务成功率	并发数	CPU 使用率	内存使用率
用户登录	<= 2.5 秒	>98%	20	＜75%	＜70%

响应时间是对用户体验影响最大的性能指标。一般来说当用户进行一项操作 2.5 秒到 3 秒后，系统没有任何反应，用户的不满情绪就会快速上升。如果业务逻辑比较复杂，理论上就不可能在 2.5 到 3 秒内完成，需要使用进度条或状态条等方式告知用户后台系统正在工作，进度到了什么情况，预计还需要多长时间可以完成。

除了软件的要求外，还应该对硬件资源进行监控，比如应用服务器的 CPU 使用率、内存使用率、带宽情况、Web 服务器的资源使用情况等，如果用户未提出要求，通常情况下，CPU 的使用率不超过 75%，内存使用率不超过 70%都是可以接受的。之所以选择这两个数值，是因为它们具有代表性。CPU 的使用率超过 75%可以说是繁忙，如果持续在 90%甚至更高，很可能导致机器响应速度大幅度降低甚至失去响应等问题。如果过低也不好，说明 CPU 比较空闲，可能存在资源浪费的问题。对于内存也存在同样的问题。

在分析并提取指标后，就可以开始建立业务模型的工作。建立业务模型需要建立在对用户行为调查的基础上，建立用户业务模型。建模是性能测试的基础。只有建立合理有效的业务模型，才能模拟出真实的系统使用情况，验证关键业务逻辑的性能，分析潜在的性能缺陷，所以建立恰当的业务模型是我们性能测试成功与否的关键。

还是以网上商城为例，用户在商城上选择一款商品完成交易，首先要登录网上商城，浏览感兴趣的商品，选定要购买的商品，在购物车中设置购买数量，下单，完成支付，如图 2-4 所示。

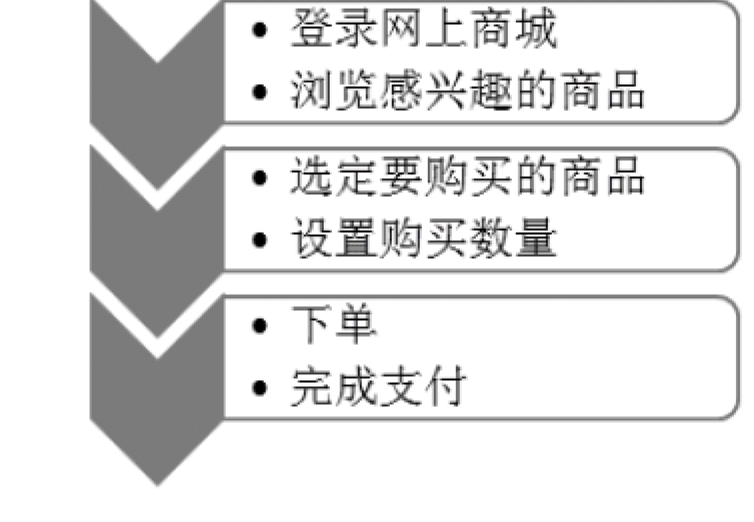


图 2-4　网上商城交易流程图

建立的业务模型如表 2-5 所示。

表 2-5　网上商城交易业务模型

步 骤 序 号	步 骤 描 述
1	使用合法的用户名和密码登录网上商城
2	浏览感兴趣的商品
3	选定要购买商品
4	在购物车中设置购买数量
5	下单
6	完成支付

经过分析测试要求与建立业务模型两步，基本上已经确定了本次测试的内容。大多数项目的性能测试分析都可以使用这样的方法。在分析与建模过程中，最重要的是要弄清楚当前测试的重点是什么，对应的业务流程是什么，就像我们做功能测试一样，性能测试也需要在客户的实际应用基础上开展，否则脱离实际的测试是无效的。

前面的两步仅是测试工程师的分析确定过程，得到测试指标与模型后，还需要项目组内其他测试工程师、开发工程师和项目经理集体进行评审，之后才可以编写对应的性能测试计划或性能测试方案，并提交项目进行审批。具体评审方法将在后面章节具体介绍。

2.2.3　分析的过程

软件测试需求分析的方法是整个软件测试需求分析过程中最重要的一环，因为测试需求就是对要测试软件的分析，以便策划和设计测试。如图 2-5 所示，软件测试需求分析是软件测试设计的基础。

软件测试需求分析过程中还有许多其他重要的环节。其中包括软件测试需求分析干系人分析、测试需求的收集与分析、测试需求的优先级排序和评审测试需求等。

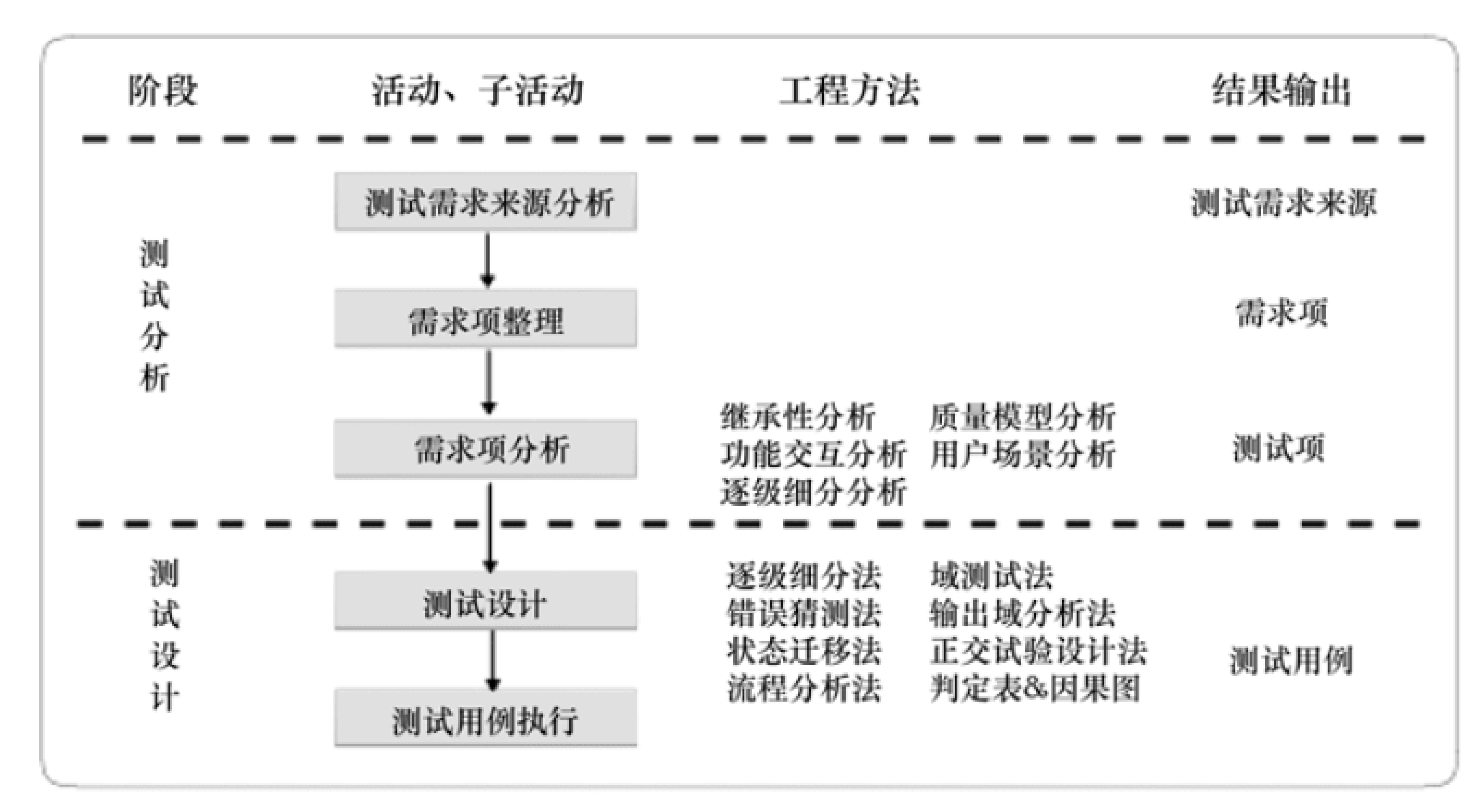


图 2-5　测试分析与测试设计关系图

1. 软件测试需求分析干系人分析

项目干系人又称项目相关利益者，是指积极参与项目或其利益会受到项目执行或完成情况影响的个人或组织。项目干系人对项目的目的和结果施加影响。项目管理团队必须识别项目干系人，确定他们的需求和期望，尽最大可能地管理与需求相关的影响，以获得项目的成功。

软件测试活动涉及的相关人员非常复杂，直接参与的有软件测试团队中的软件测试工程师、测试主管和测试经理。在一些大公司，还有专门的提供测试工具、测试方法学支持的测试专家、测试架构师团队，他们会对测试工作提出建议，甚至可能提出强制性要求。关系稍远的有同一项目团队中的开发团队成员和项目经理。再远一点的包括公司高级管理层、最终用户、支持服务工程师等。这些都是与软件测试需求分析息息相关的测试活动干系人。识别出软件测试活动的干系人，并对干系人的兴趣、影响力等进行分析，理解关键项目干系人的

需要、希望和期望。

软件测试活动干系人管理是指对软件测试活动干系人需要、希望和期望的识别，并通过沟通上的管理来满足其需要、解决其问题的过程。软件测试活动干系人管理将会赢得更多人的支持，从而能够确保项目取得成功。具体来说，干系人管理能够带来以下好处：

将会赢得更多的资源，通过软件测试活动干系人管理，能够得到更多有影响力的干系人的支持，自然会得到更多的资源；快速频繁的沟通将能确保对项目干系人需要、希望和期望的完全理解；从某种意义上来说，测试需求管理是软件测试活动干系人管理的一部分。能够预测软件测试活动干系人对项目的影响，尽早进行沟通和制定相应的行动计划，以免受到项目干系人的干扰。

通常，测试经理或测试主管需要在坚持原则的前提下与测试活动干系人之间达成共识，展开广泛而深刻的沟通，了解不同测试活动干系人对测试活动的要求，这些要求之间可能存在矛盾，这就需要测试管理团队对这些要求进行评估，并和相关干系人深入沟通、协调，最终达成一致意见。测试管理团队在测试活动中还要随时保持与各位干系人的沟通，根据实际情况对测试需求进行变更管理。测试工程师据此对测试需求进行分析，这样才能保证测试活动始终在正确的道路上前进。

在做测试需求分析干系人管理的过程中，首先要抓住最终用户，归根结底软件是为最终用户服务的。最终用户也不是呈现简单的单一状态，不同的用户有不同的诉求。明确软件的用户包括哪些人群，不包括哪些人群，哪些人群是重点用户，深刻理解他们对质量的诉求，对他们的要求要最先满足。比如赛车、第一人称射击类游戏，一方面不必考虑如何吸引只喜欢玩益智类小游戏的用户，一方面要考虑不同用户的诉求，有的用户比较看重流畅度，可以牺牲画质，有的用户则恰恰相反。这些都影响到测试需求分析。

在实际项目中，开发团队和项目经理对测试需求分析的影响是最直接也是最持久的。大部分软件测试需求分析是基于软件开发需求分析推导出来的，这就要求测试经理与开发经理、项目经理积极展开沟通，随时保持相互状态的更新。一方面，对软件开发需求进行评审，另一方面，当测试需求分析初步完成以后，也需要请软件开发团队和项目经理对测试需求说明书进行评审。在测试需求分析过程中，最易出现的问题是软件开发需求分析进度落后，造成没有足够时间展开测试需求分析工作。有效的方法是针对已经完成的软件开发需求部分尽快开展软件测试需求分析工作，不一定要等软件开发需求完全完成再开始软件测试需求分析。

公司的高级管理团队是重要的干系人。通常公司的高级管理团队不会直接具体指导测试需求分析这样相对底层的技术工作，但是在做测试需求分析的时候充分征求高级管理团队的意见，可以发挥他们对最终用户更为了解、对测试工作起到高屋建瓴的指导作用。另外，多与高级管理团队沟通，使他们了解软件质量的重要性，可以争取更多的资源，有利于进一步提高软件质量。

公司内部或外部的测试架构师、测试专家也是很重要的软件测试需求分析干系人。有利的方面是他们可以为当前的测试需求分析工作带来建设性意见或先进的工具、方法；不利的方面是可能由于他们对项目缺乏了解，而且又不能全职专注于这个项目，他们所熟悉并推崇的方法或工具可能不完全适用。比如基于模型的测试方法(Model Based Test)，这种方法把被

测系统按照状态机构建数学模型，再通过状态的转换生成测试用例。典型的测试模型有有限状态机、UML 模型、马尔可夫链和文法模型，如果被测系统不能抽象成其中某一种模型，基于模型的测试方法在这个项目中就不是最佳的解决方案。所以在处理与这些测试专家的关系时，需要注意他们推崇的方法是否适合自己的项目。

2. 需求的收集和整理

对软件测试需求干系人分析完成后，就进入原始需求的收集和整理阶段。原始需求的主要来源有 5 个：开发需求、相关法律协议和规范、以往类似项目的测试需求说明书、继承产品需求和用户原始需求。其中更多的原始需求来自开发需求、相关法律协议和规范，以及继承产品需求，尤其很多相关协议和规范是硬性规定，如果不通过相关机构的测试，软件产品就不能正式上市发行。

整理原始需求的第一步是收集跟踪所有原始需求的来源，并且制定原始需求收集表，保证团队成员可以共享相关信息。在收集原始需求来源的时候，要注意完备性，不能有遗漏。示例如表 2-6 所示。

表 2-6　原始需求收集表

原始需求来源	来源编号	文档名称
开发需求	DR001	网上商城设计需求样例 V5.DOC
开发需求	DR003	网上商城设计规格样例 V5.DOC
用户原始需求	UR001	网上商城用户意见征集表 V5.XLS
协议和规范	PR001	由于是样例，没有分析协议
继承产品需求	SR001	网上商城设计需求样例 V4.DOC
测试经验库	ER001	网上商城测试需求说明书 V4.DOC

下一步就是通过收集产品原始需求整理生产测试原始需求，可以参考表 2-7。

来源编号：与表 2-6 中的“来源编号”相同。

测试原始需求编号：编号规则为特性编码＋XXX，“特性编码”为针对开发提供的特性进行编码，可以用缩写作为编码(如 VPMN 特性，可以缩为 VPMN)，也可以顺序编号(如 R001 等)。XXX 为顺序编号，对于同一开发特性，如果有多条原始需求，可以按照顺序编号(001 开始)。

描述：对原始需求的描述，可以是来源文档中需求描述的复制，或是从测试角度提炼出来的描述。

开发特性：表示开发文档中的功能特性。

需求标识：表示原始需求在来源文档中的标识。

需求描述：表示原始需求在来源文档中的描述，如果此项与“描述”相同，可以不填写，是可选项。

表 2-7　测试原始需求

来 源 编 号	需 ：: 标 识	需 求 描 述	开发特性	测试原始需求编号	描　　述
D .001	OR_C .RT.00010	用户可以通过购物车设置需要购买的商品的数量	Cart	'art-001	用户可以通过购物车设置需要购买的商品的数量
D .001	OR_C .RT.00011	用户可以将购物车中的商品删除	Cart	'art-002	用户可以将购物车中的商品删除
D .001	OR_C .RT.00012	购物车内的商品可以与用户账号绑定，同一用户在不同客户端登录，购物车内显示的商品种类和数量不变	Cart	'art-003	购物车内的商品可以与用户账号绑定，同一用户在不同客户端登录，购物车内显示的商品种类和数量不变

产品功能不是独立的，功能之间存在交互，为了防止有交互作用的功能发生遗漏，提高功能测试的完备性，必须使用功能交互工程方法，功能交互分析是功能测试方面的分析，与测试类型分析形成互补。接下来是根据原始需求和功能交互关系进行测试需求分析。

进行分析时需要充分考虑功能之间的时序关系影响和由于共享资源产生的影响。以网上商城的购物车为例，购物车中商品的数量与商品库存量有关，这两个功能就要进行交互分析。最终生成的测试需求可以参考表 2-8。

表 2-8　测 试 需 求

来源编号	需 求 标 识	需 求 描 述	测试原始需求编号	测试需求描述	优先级	测 试 方 法
DR001	O _CART.00 10	用户可以通过购物车中默认的商品数量设置需要购买的商品的数量	ART-001	用户可以通过购物车默认的商品数量购买商品	1	功能测试
DR001	O _CART.00 11	用户可以在购物车中输入合法的商品数量并成功下单	ART-001	用户可以通过购物车设置需要购买的商品的数量	1	功能测试
DR001	O _CART.00 12	用户在购物车中设置商品数量为0或负数,系统应有友好提示	ART-001	用户在购物车中设置商品数量为0或负数,系统应有友好提示	1	功能测试

（续表）

来源编号	需求标识	需求描述	测试原始需求编号	测试需求描述	优先级	测试方法
DR001	OR_CART.00013	用户在购物车中设置商品数量为超过商品库存量，系统应有友好提示	CART-001	用户在购物车中设置商品数量为超过商品库存量，系统应有友好提示	1	功能测试
DR001	OR_CART.00014	用户在购物车中设置商品数量为非整数类型，系统应有友好提示	CART-001	用户在购物车中设置商品数量为非整数类型，系统应有友好提示	1	功能测试
DR001	OR_CART.00015	用户可以将购物车中的商品删除	CART-002	用户可以将购物车中的商品删除	2	功能测试

在这一阶段，主要测试需求的粒度，尽量保证测试需求的层次一致、粒度均匀，并且需要对测试需求的优先级进行定义。

至此，软件测试需求分析过程结束，下一阶段进入结果分析和评审过程。

2.2.4　分析结果和评审

测试工程师完成测试需求分析后，还需要进行评审。评审的内容包括完整性检查和准确性检查。完整性审查应保证测试需求能充分覆盖软件需求的各种特征，重点关注功能要求、数据定义、接口定义、性能要求、安全性要求、可靠性要求、系统约束等方面，同时还应关注是否覆盖开发人员遗漏的、系统隐含的需求；准确性审查应保证所描述的内容能够得到相关各方的一致理解，各项测试需求之间没有矛盾和冲突，各项测试需求在详尽程度上保持一致，每一项测试需求都可以作为测试用例设计的依据。

评审的形式比较多样，常用的有以下几种：

1. 相互评审、交叉评审

相互评审、交叉评审的一个例子是：甲和乙在一个项目组，处在一个领域，但工作内容不同，甲的工作成果交给乙审查，乙的工作成果交给甲审查。相互评审是最不正式的一种评审形式，但应用方便、有效。

2. 轮查

轮查又称分配审查方法，是一种异步评审方式。作者将需要评审的内容发送给各位评审员，并收集他们的反馈意见。

3. 走查

走查常常是由测试需求的负责人将测试需求在现场向团队相关人员介绍，以收集大家的意见。希望参与评审的其他同事可以发现其中的错误，并能进行现场讨论。这种形式介于正式和非正式之间。

4. 小组评审

小组评审就是通过正式的小组会议完成评审工作，是有计划的和结构化的评审方式。评审定义了评审会议中的各种角色和相应的责任，所有参与者在评审会议的前几天就拿到了评审材料，并对材料进行了独立研究。审查和小组评审很相似，但更为严格，是最系统化、最严密的评审形式，包含制订计划、准备和组织会议、跟踪和分析审查结果等。

评审的人员组成需要仔细遴选，在正式评审小组中，一般存在多种角色，包括协调人、作者、评审员等。评审员需要精心挑选，保证不同类型的人员都要参与进来，通常包括开发经理、项目经理、测试经理、系统分析人员、相关开发人员和测试人员等。

2.3　软件测试需求管理的内容

本节讨论的内容很重要。前面的内容解释了软件测试需求管理相关的基本概念、分析方法、过程等。本节介绍软件测试需求管理的内容和相关的概念及要点。

软件需求管理的任务是获取、文档化和评审用户需求，并对用户需求的变更进行控制和管理。软件测试需求管理的内容在业界有不同的说法和详细程度。但常见的认识是软件测试需求管理主要包括几个要素：定义测试需求、测试需求分析、测试需求筛选、测试需求处理等。另一种思路是从关注几个方面的问题来分析软件测试需求管理的内容：

1. 定义测试需求

在项目进展过程中，在完成用户需求调查后，首先对《用户需求说明书》或其他方式说明的软件产品设计说明书进行细化，对比较复杂的用户或设计需求进行建模分析，以帮助软件开发人员更好地理解需求。在完成需求的定义及分析后，需要将此过程书面化，要遵循既定的规范将需求形成书面文档，我们通常称之为《需求分析说明书》。同时还要明确对测试工作有什么特别的需求和限制。比如软件服务外包项目中的测试项目常常有预算、人员、时间、工具、测试用例等具体要求。这些信息是所有测试需求的参考依据，也是测试需求管理的第一步。

2. 确认测试需求

需求确认是需求管理过程中的一种常用手段。确认有两个层面的意思，第一是进行系统需求调查与分析的人员与客户间的一种沟通，通过沟通从而对需求不一致的地方进行剔除；另一层面的意思是指，对于双方达成共同理解或获得用户认可的部分，双方需要进行承诺。

软件测试需求的需求确认就是针对上一步的信息加以分析，将测试的特别需求列出来，并和相关各方负责人确认。

3. 建立测试需求状态

何谓需求状态。顾名思义，状态也就是一种事物或实体在某一时刻或时间点所处的情况，此处要讲的测试需求状态是指用户或软件需求的一种状态变换过程。

为什么要建立测试需求状态？在整个测试生命周期中，存在着几种不同的情况，在测试需求调查人员或系统分析人员进行需求调查时，客户存在的测试需求可能有多种，一类是客户可以明确且清楚提出的测试需求；一类是客户知道需要做些什么，但又不能确定的测试需求；另一类是客户本身可以得出这类测试需求，但需求的业务不明确，还需要等待外部信息。还有就是客户本身也说不清楚的。如果是自主研发的软件，那就需要软件架构和功能设计人员提供信息或确认需求。测试需求状态就会根据他们的需求状态而定。

对于软件功能设计或用户设计的需求，在软件开发进展过程中，对于每个功能或模块，存在着以下几种情况：

- 必须要完成的；
- 有时间争取一定完成的；
- 有可能要取消的；
- 有的因为不明确而可以延后的，同时可能转换为被取消的需求；
- 与客户或项目经理经过沟通或确认的，此处有两种情况，一类是确认双方达成共识，另一种情况是还需要再进一步沟通。

4. 测试需求评审

邀请团队内部和外部专家，以及了解测试需求的用户(包括客户和最终用户)一起评审测试需求，尽最大努力使记录下来的测试需求能够正确无误地反映用户的真实意愿。测试需求评审之后，开发方和测试方的责任人对测试需求作书面承诺。具体的同行评审详见需求评审章节。

测试需求评审的规程与其他重要工作产品(如系统设计文档、源代码)的评审规程非常相似，主要区别在于评审人员的组成不同。前者由开发方和客户方的代表共同组成，而后者通常由项目团队的测试人员以及与所评审测试对象相关的人员组成。

测试需求评审比较费脑子。有时刚开始评审时，大家都比较认真，越到后头越松散。当测试需求文档或评审会议很长时，几乎没人能够坚持到最后。会议主持人事先要强调测试需求评审的重要性：认真评审一小时可能会避免将来数十天的“返工”，让大家足够重视。评审组长还要设法避免大家在昏昏沉沉中评审。如果评审时间比较长，建议每隔两小时休息一次。另外，如果系统比较大，也可以细分成不同的部分分别进行，严格控制每一次评审的文档规模及持续时间。

5. 测试需求责任制

需求责任制，是指测试团队负责人要把某一项测试需求分派给指定负责人，同时承诺应

交付的测试结果和其他交付物。

6. 测试需求跟踪

进行测试需求跟踪的目的是为了建立和维护从用户需求或设计人员开始到测试之间的一致性与完整性。确保所有的编程和技术实现是以用户需求或软件设计人员为基础，确保需求实现全部覆盖，同时确保所有的输出与用户需求的符合性。

需求跟踪有两种方式，正向跟踪与逆向跟踪。

- 正向跟踪以用户需求为切入点，检查《用户需求说明书》或《需求规格说明书》中的每个需求是否都能在后继测试工作产品中找到对应点。
- 逆向跟踪检查设计文档、代码、测试用例等工作产品是否都能在《需求规格说明书》中找到出处。

正向跟踪和逆向跟踪合称为“双向跟踪”。不论采用何种跟踪方式，都要建立与维护《需求跟踪矩阵》。很多人有这样的误解：如果依照“需求开发-系统设计-编码-测试”这样的顺序开发产品，由于每一步的输出就是下一步的输入，所以不必担心设计、编程、测试会与需求不一致，因此可以省略需求跟踪。那么需要指正的是，按照软件生命周期严格线性顺序的开发模型并不能保证各个开发阶段的工作产品与需求保持一致。因为开发者是人而不是机器，而且大多数开发人员也都深有体会。

由于人们的表达能力、理解能力不可能完全相同，人与人之间的协作很难达到天衣无缝的境界，而使用需求追溯本身也是一种传递的过程。

需求追溯本身并不是一件复杂的事，而是我们本身的一种理念在支配着我们。也许就有人认为这本身就是在浪费时间，当需求或工作产品发生变更时，开发人员要及时更新需求跟踪矩阵。然而没想到的是，如果后来再花时间来做同样的事情，将会付出更多。

2.3.1　变更管理

事实上软件开发本身的需求也会有变化，可能因为客户的需求，也可能因为市场的需求，还可能因为技术或非技术的其他原因。需求的变化问题是每个开发人员、每个项目经理都必须面对的问题，也是最头痛的问题。一旦发生需求变化，就不得不修改设计、重写代码、修改测试用例、调整项目计划等。需求的变化好比万恶之源，为项目的正常进展带来不尽的麻烦。解决的办法是有效地对需求进行管理，使需求在受控的状态下发生变化。需求管理就是要按照标准的流程来控制需求的变化。

需求变化通常是渐变问题，这种渐变很可能是客户与开发方都没有意识到的，当达到一定程度时，双方才发现，但纠正已为时过晚。

而这些变动会导致相关软件开发活动以及软件产品的变化，如软件设计、编码和测试等。而且有时软件需求的一点变动往往会导致其他软件产品大幅变动。因此，软件测试需求也要有应对变更的测试需求变更管理。

1. 测试需求变更管理的必要性

随着测试工作的展开，软件测试需求并不是一成不变的。需求变更本是正常的，并不可怕，可怕的是需求的变更得不到控制。软件需求的变化，软件测试干系人的期望值和人员、进度、预算变化，均有可能引发软件测试需求随之改变。这就需要对软件测试需求实施变更管理。

对测试工作而言，测试需求变更管理是个相对被动的过程，软件需求发生了变更，测试需求必须随之变化。软件需求一旦发生变化，就要对需求跟踪表进行维护，启动配置管理过程，与软件需求变更相关的内容进行同步变更。

对需求的变更，从某种程度说，也就是项目的范围发生了变化。而需求同时又是项目进行的基础，是非常重要的基石。通常对于需求的变更需要客户与开发方共同参与，包括负责人及市场人员。当然，我们需要根据变更的内容来灵活运用。

如果孤立地看软件测试需求变更，大致就是提出变更申请-修改并评审变更-完成变更这样的过程。但软件测试需求变更管理并不是单纯的一系列数据库记录，或只是备忘录而已。宏观地看，如果把所有的软件测试需求变更完善地管理好，并加以分析和应用，可以发挥两个重要用途：一是保证软件测试工作可控；二是通过对软件测试需求变更度量分析，打造并不断提高测试团队对测试工作，尤其是快速多变的测试工作的驾驭能力。

2. 测试需求变更管理的意义

可使用测试需求跟踪矩阵对测试需求进行管理。为了保证软件测试工作可控，测试经理要充分了解软件测试需求变更的触发因素，衡量变更实施对项目的冲击，决定是否要修改。软件测试需求的变化是否严重到必须立刻对当前的测试工作实施变更；问题的修改是否很复杂，是否会牵扯很多方面。根据上面的介绍，软件测试需求主要来自软件开发需求，软件测试需求的变更大致可以归为两类：一类是发生变更的软件开发需求对应的软件测试需求；另一类是关联信息，即某个功能发生变更后，对与之有交互关系的软件测试需求也需要进行变更，这类信息通常不可能由提出软件开发需求变更的人提供，而需要变更实施者和审核者结合多方面信息分析提出。

实施软件测试需求变更管理的更重要且更有意义的作用就是对变更进行度量分析，打造并提高测试团队的应变能力。在测试工作过程中，对变更进行分析，可以很好了解项目当前的状态；在每个迭代周期结束的时候进行回顾，分析软件测试工作中变更的产生原因和解决方法，并评估当时采取的措施是否合理有效，再遇到类似变更是否有更有效的措施，促使测试团队的应变能力不断得到提高。

3. 软件测试需求变更的主要任务

- 提出变更；
- 分析变更的必要性和合理性，确定是否实施变更；
- 记录变更信息，填写变更控制单，提交变更申请；
- 做出更改，并交上级审批；
- 修改相应的软件测试工作，如更新测试用例等，确定新的版本；

- 评审后，正式发布新版本的软件测试需求说明书。

软件测试需求变更申请可以由测试团队内部发起，也可以由外部提出。可触发软件测试需求变更的原因有很多，最常见的是软件开发需求变更。然而，项目的进度、人力资源、质量要求或预算等因素发生变更都可能触发软件测试需求变更。

变更控制不应该只是软件开发过程应该考虑的事情，随着软件产品的开发和时间的推进，用户会提出越来越多的新需求，甚至在交付软件产品的最后阶段用户还会有不同的需求，因此需求变更的管理应贯穿于整个项目生命周期的全过程。

为了使变更对项目的影响降到最小，应采取适当有效的变更控制策略，确定选择、分析和决策需求变更的过程，所有的需求变更都需遵循此流程。对需求变更的处理应该分以下几个步骤：提出变更、审核变更、实施评估、确认验证变更并监督变更过程。

4. 建立测试需求管理模型

需求建模是表达需求的一种形式，是对需求的描述与阐释，它使用标准语言或思维导图。比如可以利用类似积木的概念来建模，最大的好处是大家可以直接根据需求，轻易地反复修改需求模型。可以使大多数人快速地理解，也可以用在测试需求管理实践中。

需求建模的目的是要消除人际沟通随意性很强的弱点，所以需要致力于将沟通标准化、自动化、准确化，而且责任到人。具有可测试、可验证性的特点。建模的过程就是通过测试需求的特点和要求进行分析，以建模标准为基础进行准确、完备和有效的阐述，以确保客户和开发方都能准确无误地理解。

5. 软件测试需求变更实例

仍以网上商城购物车为例。根据用户体验调查，项目团队决定对购物车中商品的删除功能实施变更。原始设计是在用户在购物车中执行删除操作后，商品即被移出购物车，不能再被召回。从用户体验和商业角度分析，部分用户把购物车中的商品删除后，会出现后悔的情绪，希望尽快找回删掉的商品，这就需要设计“后悔”功能。于是新设计的购物车的删除功能只是将商品暂时移出购物车，并且在购物车的页面添加一个已删除商品的栏目，顾客可以将已删除的商品恢复到购物车中。

针对这一软件开发需求变更，负责购物车模块的测试工程师应对与之对应的软件测试需求进行审核。很明显，先前的软件测试需求已经不足以覆盖变更后的软件开发需求。

该软件测试工程师应向软件测试经理提出软件测试需求变更申请，在得到肯定的批复后立即着手软件测试需求的修改工作。通过变更后的软件开发需求，可以推导出至少两个软件测试需求，一个是将购物车中的商品删除，商品进入购物车的已删除商品栏目；另一个是将购物车的已删除商品栏目中的商品恢复到购物车中。相应的软件测试需求文档(见表 2-9)更新如表 2-10 所示。

在更新软件测试需求后，需要对新的软件测试需求进行评审，评审通过后即可正式更新软件测试需求说明书，并向测试团队和整个项目团队发布新版本的软件测试需求说明书。

表 2-9　软件开发需求变更前的软件测试需求

来源编号	需求标识	需求描述	测试原始需求编号	测试需求描述	优先级	测试方法
DR001	OR_CART.0011	用户可以将购物车中的商品删除	CART-002	用户可以将购物车中的商品删除	2	功能测试

表 2-10　软件开发需求变更后的软件测试需求

来源编号	需求标识	需求描述	测试原始需求编号	测试需求描述	优先级	测试方法
DR001	OR_CART.0001	用户可以将购物车中的商品删除并恢复	CART-002	用户可以将购物车中的商品删除，被删除商品进入购物车的被删除商品列表	2	功能测试
DR001	OR_CART.0001	用户可以将购物车中的商品删除并恢复	CART-002	用户可以将购物车被删除商品列表中的商品删除恢复到购物车中	2	功能测试

2.3.2　状态管理

状态是事物或实体在某个时刻或时间点所处的情况，此处讲的测试需求状态是指软件测试需求的一种状态变换过程。在软件测试需求的生命周期中，软件测试工程师对软件测试需求做调查发现，软件测试需求可能有几类：一类是明确且清楚地提出软件测试需求；另一类是虽然知道大致需要做什么，但又不能确定软件测试需求；还有一类是需要等待外部信息来进一步明确的软件测试需求；最后一类是没有明文说明、隐含的软件测试需求。对这些软件测试需求，在测试工作进展的过程中，可能存在若干种状态：

- 只知道大致需求，具体细节还有待细化；
- 已经初步确定，等待评审；
- 已经确定，并经过团队评审，在可预期未来不会发生变更；
- 已经评审完毕正在进行设计、实现测试用例的测试需求；
- 完成设计、实现测试用例的测试需求。

显然，软件测试需求的这些状态间可以互相转换，如图 2-6 所示。在不同风格的软件测试管理方法或工具中，定义的软件测试需求状态也不尽相同，但只需要有符合实际测试项目需求管理要求的状态分类进行管理就可以实施。本书仅列举业界用过的几种需求分类状态供参考。

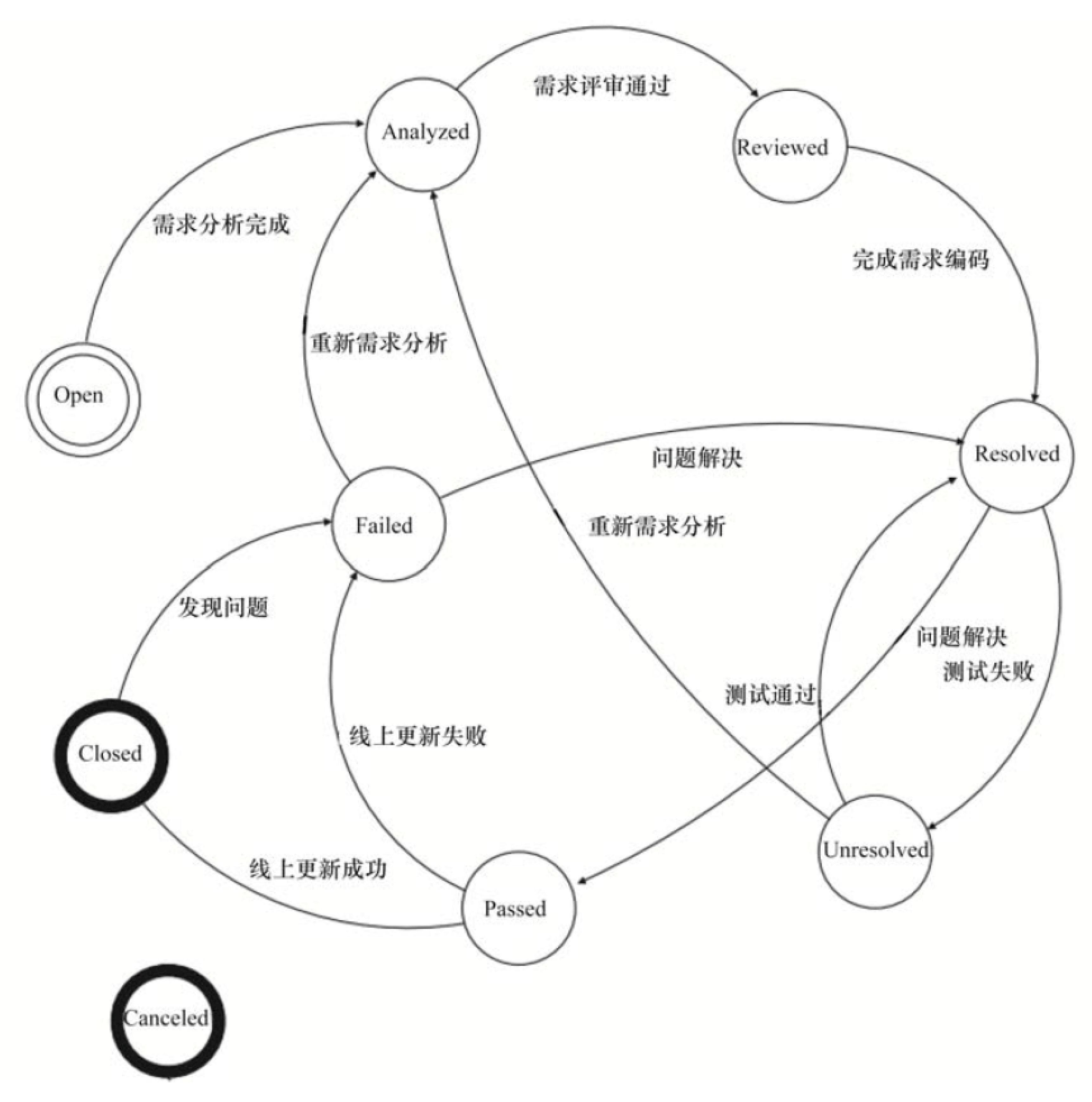


图 2-6　测试需求状态转换

(1) Open：对于原始需求或接收到的正式需求，但未正式进行需求分析之前的需求状态统一定义为“Open”状态。

(2) Analyzed：对需求状态为“Open”的需求，已完成需求分析过程，在还未正式通过需求评审前，状态统一定义为“Analyzed”。

(3) Reviewed：对需求状态为“Analyzed”的需求，已正式通过需求评审，在还未完成测试或测试结果为不合格之前，状态统一定义为“Reviewed”。

(4) Resolved：对需求状态为“Analyzed”或“Reviewed”的需求，已完成需求设计和编码，且已通过单元测试，状态统一定义为“Resolved”。

(5) Passed：对需求状态为“Resolved”的需求，如果已通过正式测试，状态统一定义为“Passed”。

(6) Unresolved：对需求状态为“Resolved”的需求，如果未通过正式测试，状态统一定义为“Unresolved”状态。

(7) Closed：对需求状态为“Resolved”的需求，如果需求已正式上线商用，且得到客户和项目团队的共同认可，状态统一定义为“Closed”。

(8) Cancel：当原定义的某些需求被取消时(包括上线前取消和上线后取消)，需求状态统

一定义为“Cancel”。

(9) Failed：对需求状态为“Closed”的需求，当需求在上线商用后发现问题或存在缺陷，需要进行修正时，需求状态统一定义为“Failed”。

2.3.3　文档版本管理

软件测试需求文档的版本管理是软件测试需求管理的基础，借此可以使得同一软件测试需求文档被测试团队中不同的人员编辑，并且记录下每次编辑的增量，必要的情况下还可以回滚到某个版本。

当下流行的文档版本管理方式是通过带有安全授权机制的专业软件管理软件，如惠普 ALM 工具，对测试需求进行管理。软件测试需求文档集中存放在服务器上，经系统管理员授权给不同的用户群组，不同的用户群组拥有不同的读写、只读或创建等权限。软件测试工程师通过签出(check out)和签入(check in)的方式更新系统中的软件测试需求文档，只读用户则可以通过系统查看软件测试需求文档的最新或某个历史版本。未经授权的用户则无法访问服务器上的文件。这样可保证所有具有授权的用户都能看到统一的最新版本的软件测试需求文档，避免有的用户基于较老的版本工作，做无用功。

软件测试需求文档升级管理：每次有用户签入(check in)时，在服务器上都会生成新的版本，记录本次操作的用户、操作时间等信息，并且保留旧的版本，使得所有的更改都有历史记录可查，之前的任一版本都可以随时调阅出来进行阅读或编辑。即使有用户进行错误的编辑，甚至删除软件测试需求文档的部分或全部内容，也依旧可以保证在很短的时间内恢复回来。

类似惠普 ALM 这样的专业管理软件对多用户操作有着良好的支持。不同的用户可以在同一时间对同一份软件测试需求文档进行更新，如果有不同用户对同一段内容进行编辑(即发生“冲突”)，可以向用户发出友好的提示，甚至在一定程度上自动解决部分冲突。这样的功能为多人团队尤其是异地团队合作提供了极大方便。

如前所述，软件测试需求文档有了良好的版本管理之后，用户可以同时调阅两个甚至若干个历史版本进行比较、分析。这个功能是下一节软件测试需求跟踪管理的基础。

2.3.4　跟踪管理

软件测试需求跟踪是指跟踪软件测试需求使用期限的全过程。实施软件测试需求跟踪为我们提供了由软件测试需求到完成软件测试工作整个过程范围的明确查阅能力。软件测试需求跟踪的目的是建立与维护“软件测试需求-测试用例设计-设计用例实现-测试用例执行”之间的一致性，确保所有的测试工作交付物符合软件测试工作的初衷，如图 2-7 所示。

需求跟踪包含的正向跟踪和逆向跟踪合称为“双向跟踪”。不论采用何种跟踪方式，都要建立与维护需求跟踪矩阵。需求跟踪矩阵保存了需求与后续开发过程输出的对应关系。矩阵单元之间可能存在“一对一”、“一对多”或“多对多”的关系。表 2-10 是一张简单的软件测试需求跟踪矩阵示例。

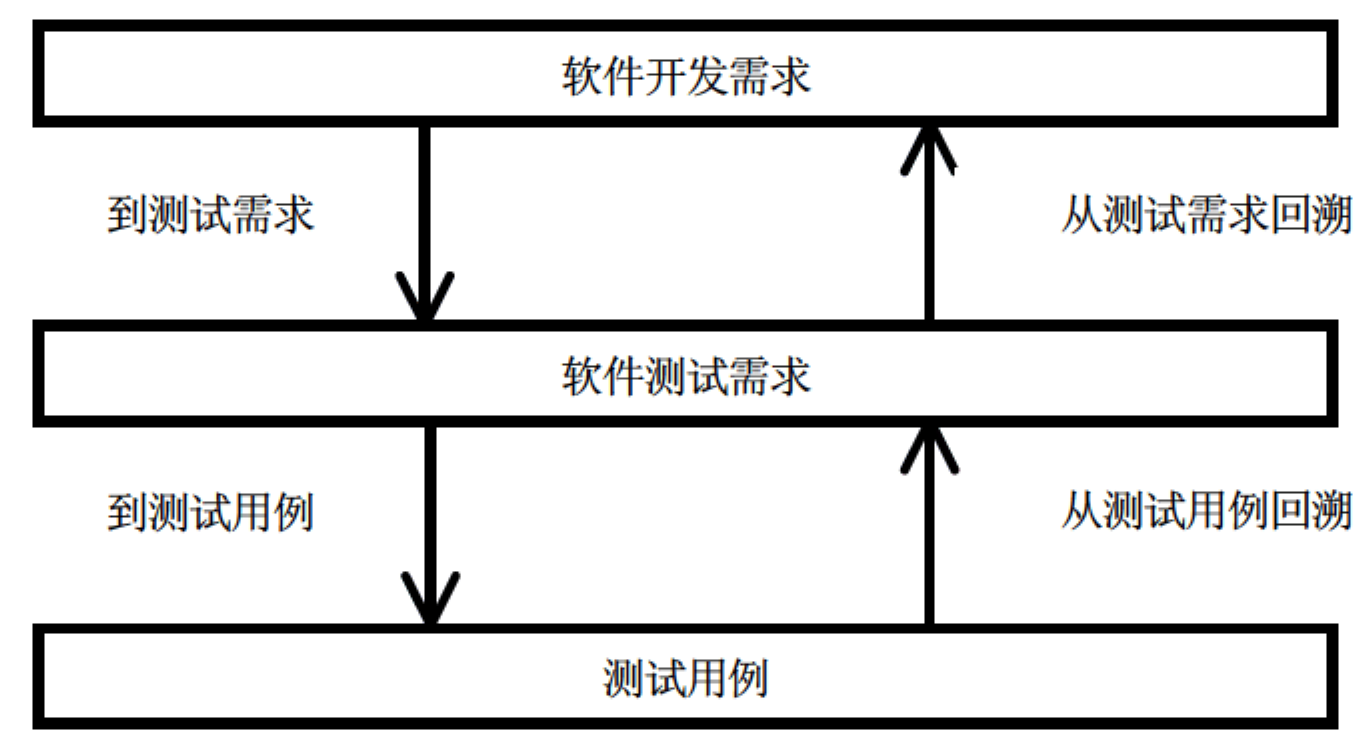


图 2-7　软件测试需求跟踪

表 2-11　软件测试需求跟踪矩阵示例

需求编号	需求标识	测试需求编号	测试需求描述	测试用例 ID	测试用例
OR_CART.00011	用户可以在购物车中输入合法的商品数量并成功下单	CART-002	用户可以通过购物车设置需要购买的商品的数量	TC_CART.00001	当商品库存不为 0 时，用户通过购物车设置需要购买的商品的数量为 1，验证可以成功下单
OR_CART.00011	用户可以在购物车中输入合法的商品数量并成功下单	CART-002	用户可以通过购物车设置需要购买的商品的数量	TC_CART.00002	当商品库存不为 0 时，用户通过购物车设置需要购买的商品的数量为 5(或其他小于商品库存的数量)，验证可以成功下单
OR_CART.00011	用户可以在购物车中输入合法的商品数量并成功下单	CART-002	用户可以通过购物车设置需要购买的商品的数量	TC_CART.00002	当商品库存不为 0 时，用户通过购物车设置需要购买的商品的数量恰为商品库存的数量，验证可以成功下单
OR_CART.00011	用户可以在购物车中输入合法的商品数量并成功下单	CART-002	用户可以通过购物车设置需要购买的商品的数量	TC_CART.00002	当商品库存不为 0 时，用户通过购物车设置需要购买的商品的数量为比商品库存大 10 的数量，验证不可以成功下单，并有用户友好的提示

软件测试需求变更(增、删、改)都在表中记录，为测试团队提供软件测试需求跟踪能力。使用软件测试需求跟踪矩阵对软件测试需求进行跟踪的好处如下：通过跟踪软件测试需求的后续测试用例信息可以帮助确保所有软件测试需求被实现，没有遗漏；通过对软件测试需求变更影响分析的跟踪，在对软件测试需求进行增、删、改等变更时可以确保与之对应的测试用例也进行必要更新，而不被忽略；及时可靠地对软件测试需求进行跟踪，使得维护时能正确、完整地实施变更，从而提高生产效率。

当代软件测试工程推荐使用专业化的软件管理系统对测试需求进行管理。惠普的 ALM 系统就具备完善的软件测试需求管理功能，而且 HP ALM 系统更进一步提供了从软件开发需求到软件测试需求，再到软件测试用例这一完整信息链的跟踪管理能力，大大提高了软件研发的专业程度，对提高软件的质量有积极的意义。

2.4　惠普测试需求管理解决方案

用于现代应用测试的软件解决方案有惠普应用生命周期管理(ALM)工具，这是一款应用测试软件，可提供统一的存储库、一致的用户体验和可定制的仪表板(Dashboard,ALM 工具里的一个组件)，支持从单一存储库管理整个应用生命周期，包括从需求开始到准备交付为止的整个过程。跨项目报告和资产共享功能可在各个项目之间保持统一的流程，以帮助跟踪和报告项目进度和共享资产。仪表板和报告功能则通过使用企业关键绩效指标(Key Performance Indicator，KPI)分析既定里程碑的相关数据，支持跟踪整个生命周期的应用就绪状况。

测试需求管理相关模块

如图 2-8 所示，惠普应用生命周期管理流程分为以下几个阶段：

(1) 指定版本：制订一个发布周期管理计划，更高效地管理应用程序发布和周期。追踪应用程序发布，并根据计划确认发布是否正常。

(2) 指定需求：分析应用程序并确定需求。可以跨多个发布和周期管理需求，并在需求、测试、缺陷之间实现多维追踪。惠普应用生命周期管理为需求覆盖和关联到质量评估及商业风险中的缺陷提供实时可见的功能。

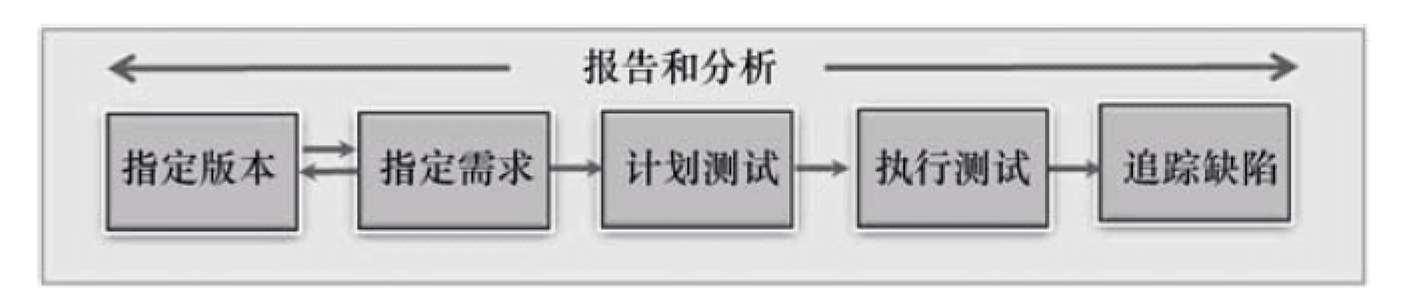


图 2-8　惠普应用生命周期管理流程图

(3) 计划测试：创建基于需求的测试计划，惠普应用生命周期管理系统为手动测试和自动测试提供了知识库。

(4) 执行测试：创建测试集，完成测试运行。惠普应用生命周期管理支持健壮测试、功能测试、回归测试及更高级测试。根据计划来执行测试，从而识别和解决问题。

(5) 追踪缺陷：报告在应用程序中检测到的缺陷，跟踪修复进程。分析缺陷和缺陷趋势，帮助做出合理的“执行/不执行”决策。惠普应用生命周期管理系统支持完整的缺陷生命周期——从初始问题检测到修复缺陷以及确认缺陷修复。

惠普应用生命周期管理系统有 11 个模块，如图 2-9 左侧边栏所示。系统中的和测试需求有关的功能菜单如下：

Requirements：指定需求。

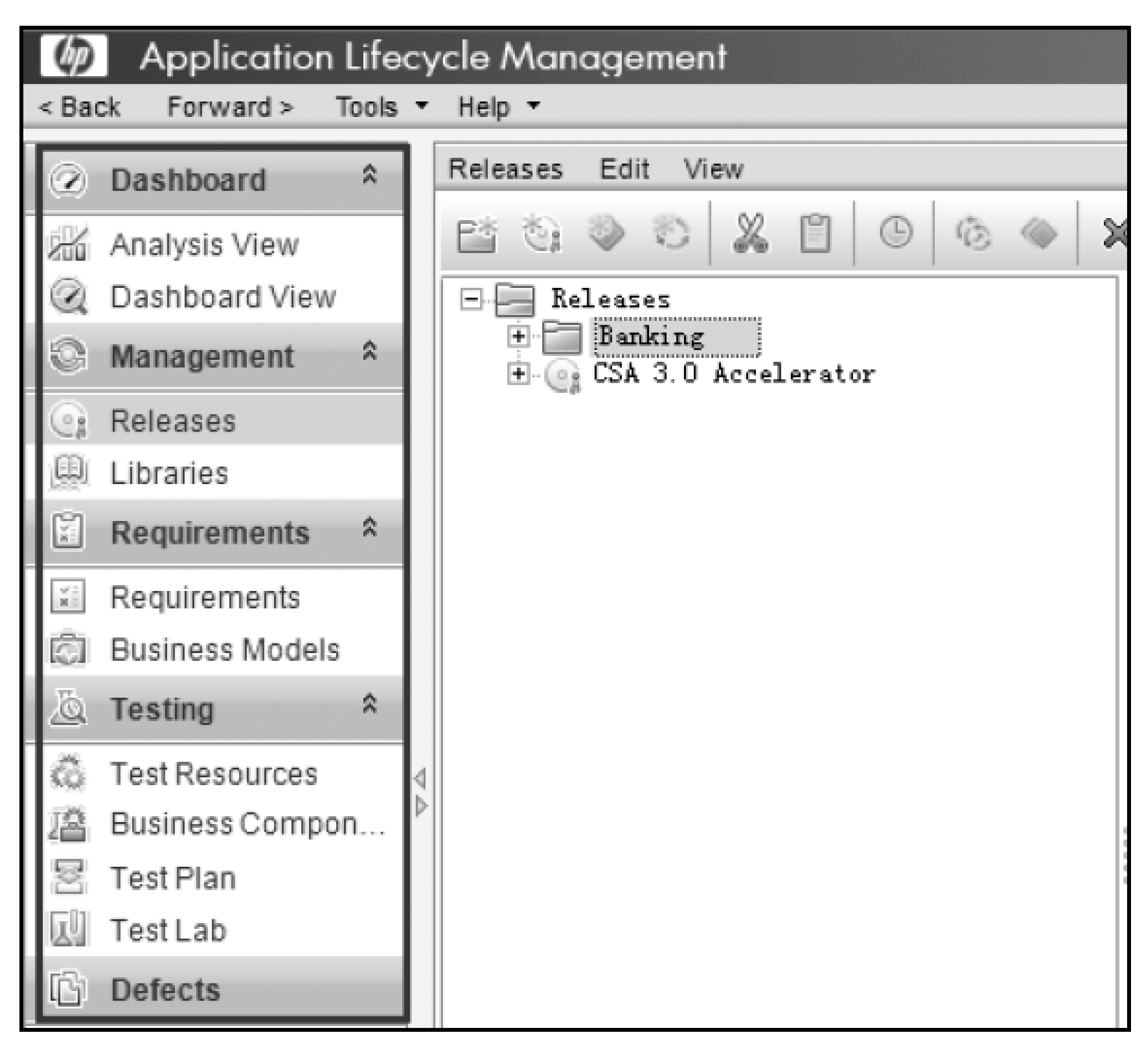


图 2-9　惠普应用生命周期管理系统的测试需求相关模块示意图

习题与思考题

1. 软件测试需求管理的定义？
2. 测试需求分析主要有哪两项主要任务？
3. 阅读并理解 2.2.2 节的内容，说说软件测试需求分析的方法和步骤？
4. 说明软件需求评审的内容，包括评审的方法、过程以及注意事项？
5. 软件需求变更的原因有哪几种？
6. 请说明软件测试需求的状态，我们通过哪些工作来保证软件测试需求状态之间的转换。
7. 惠普应用生命周期管理包括哪几个流程？

第3章　软件测试团队管理

无论软件研发和测试的技术、工具或方法多么完美，最终都要由人来执行。人是整个软件研发和测试过程中最重要的组成部分。测试过程中的任何测试活动最终都需要测试人员的参与，因此测试团队管理至关重要。测试团队所有成员的个人技能也会直接影响软件测试的最终结果，进而影响软件产品的发布质量。本章围绕测试团队的管理和建设，从组织结构、角色分工、团队文化、测试人员技能要求和绩效考核、交流沟通等方面加以分析和说明。作为应用型软件人才，学习完本章后，应具备以下能力：了解测试团队的组成以及测试工程师的分类和分工，测试工程师能够根据产品或项目的要求完善自身的业务能力，测试主管和测试经理等测试管理人员可以根据产品或项目的要求组建合理高效的测试团队，并能根据形势的发展对现有的测试团队进行调整。

3.1　重视测试团队的管理与建设

测试团队的管理与建设有很多需要考虑的因素和待解决的问题。首先要明确其重要性和必要性。接着要探讨最佳实践。大学生在校期间就应该意识到为什么重要，在实训期间也要有意识地学习怎样在团队环境里和别人协同工作。

3.1.1　重要性和必要性

人是测试工作中最有价值也是最重要的资源，没有具备测试技能的、综合能力强的、积极向上的测试团队，就不可能完成高质量的测试工作。然而，在软件开发领域有一种非常普遍的习惯，那就是让那些经验最少的新手、没有效率的开发者或不适合干其他工作的人去做测试工作。这绝对是一种目光短浅的行为，对系统进行有效测试所需的技能绝对不比进行软件开发需要的少，事实上，测试人员会遇到很多挑战，包括许多开发者不可能遇到的问题。

一大挑战是：测试时间总是不够。测试人员经常发现自己的处境是要测试的总是比有时间测试的工作量多得多、很难决定最佳有效测试范围、没有时间按部就班发挥测试最高水平、缺乏需要的软件开发功能需求文档、需求经常变动等。测试团队经常面临至少 10 种挑战。需要有团队力量的支持和理解，团队成员间的互助和理解才有可能减少这些困难，克服困难，找到解决方案，保证完成任务。这 10 种挑战包括：

- 测试人员被认为低人一等。
- 测试时间永远不够。
- 缺乏简单易用的测试辅助工具。
- 缺乏具体通用的测试技术。

- 很难清楚了解用户需求和期望。
- 缺乏可明确衡量测试质量达标的度量。
- 很难确定测试实例是否执行完毕。
- 很难找时间开发自动化测试工具。
- 测试所需文档经常不全。
- 很多任务要同时考虑，很难保质保量做好每一件。

3.1.2　专业的测试团队管理系统与工具

专业的软件测试系统集中测试相关所有专业信息于一体，是目前最为推荐的沟通方式。当前专业软件测试系统已经成为专业软件项目管理系统中的一部分，集成了产品功能管理、进度管理、测试计划、测试用例、测试结果、缺陷和测试报告等模块。从记录产品的功能开始到根据功能点设计测试用例、执行测试用例、记录缺陷生成测试报告一气呵成，完全都在专业项目管理系统中完成，测试工程师、开发工程师和项目经理通过系统进行有效沟通。

专业软件项目管理系统的优点是提供软件项目优化的专业信息，为团队提供统一的沟通风格，减少歧义，提高工作效率。缺点是门槛高，掌握并熟练使用时间成本较高，不同专业软件项目管理系统间存在较大差异，更换不同的系统，代价较大。

惠普提供的 HP ALM 工具是业界领先的专业项目管理系统，在本书的其它章节会有详细介绍和实训内容，学员应认真学习并多动手实践，切实增强工作实践能力。

3.1.3　软件测试团队管理最佳实践

软件产品开发是集体智慧和工作的结晶。软件通常由众多组件、功能、模块等构成。每部分都是由一位或多位软件工程师设计、组合、编码或测试后实现的。团队的技术水平、精神面貌、纪律、尽职程度等决定着产品最终质量。但是组建的测试团队常常在软件开发项目结束后就有很大人员变动。这种人才流失会对公司和团队接下来的项目有影响。

1. 测试团队管理建议

业界有很多公司对测试团队的管理非常重视，他们根据实践提出以下 10 项建议：

- 明确测试人员职责。
- 培养人才梯队。
- 设立“back up”体系。
- 确保测试人员有职业发展渠道。
- 打造共赢的团队文化。
- 形成学习和培训机制。
- 公平和透明的绩效考核标准。
- 经常和员工交流。
- 奖惩分明。
- 领导以身作则。

3.2　测试团队的组织管理

软件测试组织是指软件测试团队的构成、组织结构及形式。在早期的软件开发过程中，并没有专职的软件测试工程师和软件测试团队，通常由开发工程师甚至客户自己来实施软件测试工作。随着软件规模越来越庞大，结构日趋复杂，软件测试也逐渐在更加关键的位置发挥作用，因为一旦软件缺陷引发事故，损失会变得越来越不可承受。所以，惠普、微软和 IBM 等国际一流软件企业开始设置专职的软件测试工程师，进而组建越来越专业的软件测试团队，将更加严谨的测试活动融入当代软件开发过程中。

随着软件测试行业的发展，越来越多的工程师成为专职的软件测试工程师。他们不断在软件测试理论、实践以及工具方面进行归纳、总结和创新，不断提升软件测试的理论水平，软件测试在当代软件开发中占有越来越重要的地位。从社会发展的角度讲，软件测试的专业化也是社会分工的结果。

在经典的项目经理、开发团队和测试团队的“三驾马车”形式的软件研发团队中，软件测试团队担当着非常重要的责任。软件测试团队的组织架构可以从多个角度进行诠释。

3.2.1　常见测试团队的组织结构

如前所述，软件测试团队的成员有不同的分工、分担不同的角色，这些成员在一起并不是一盘散沙，而是有机的组织结构，各司其职互相支持，这样才能发挥更大作用。

当代软件测试团队的组织结构通常由测试管理团队和测试工程师团队组成。测试管理团队自上而下一般分别设立测试经理和测试主管的职位，部分企业也设置测试总监这样更高级的测试职位。他们负责软件测试的计划、预算、与其他团队沟通等管理相关的工作。测试工程师团队的成员则主要负责测试的具体技术工作，包括设计、执行测试用例，书写测试报告，报告软件缺陷等。后面将会对这些职位展开详细介绍。常见测试团队的组织结构如图 3-1 所示。一名测试经理通过若干名测试主管管理测试团队。

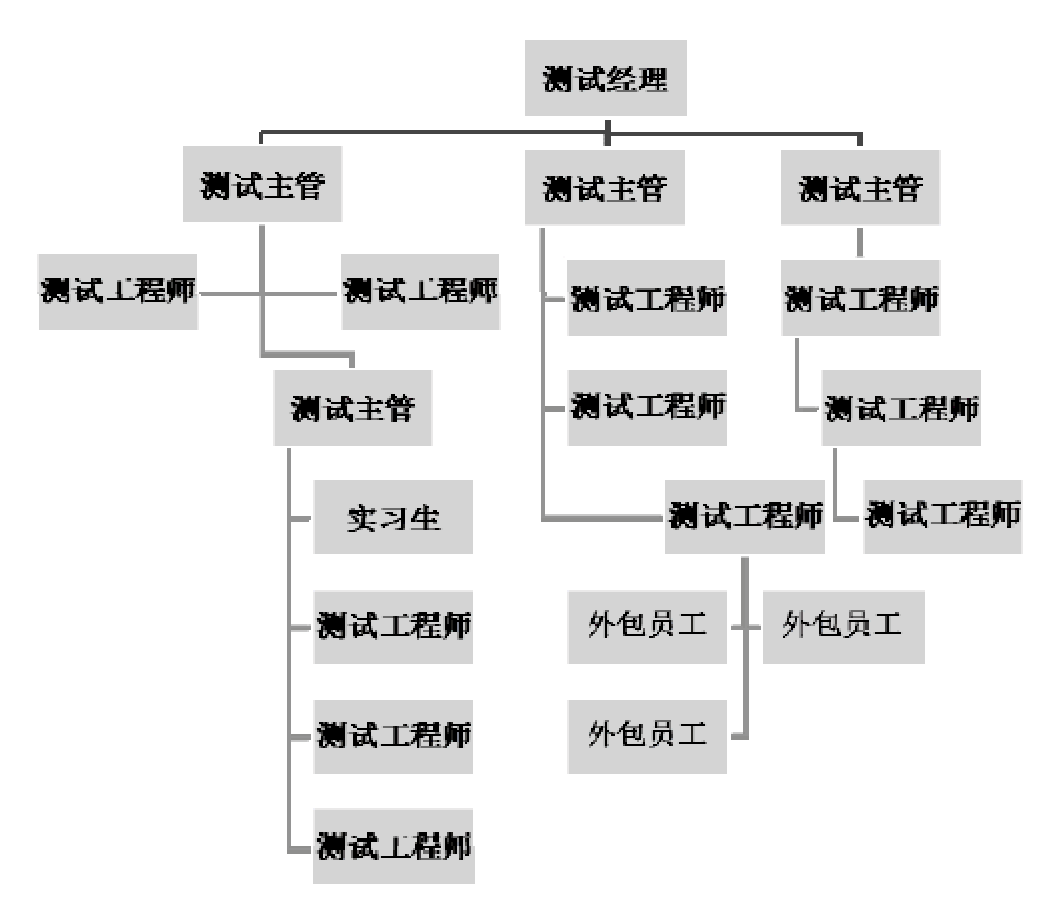


图 3-1　常见测试团队的组织结构图

3.2.2　软件测试组织的专业分工

测试团队的构成及规模根据实际项目资源和需求决定。测试工作必须有人做，但未必需要单独成立测试团队。开发人员也应该懂测试和实施一些测试。比如他们通常需要测试自己的代码，即他们需要进行白盒测试。如果一个项目只有一个人，那么这个人需要扮演不同的角色。既做项目经理，又做开发工程师和测试工程师。测试人员也常常是不同测试角色的通

用名称。

在业界很多软件研发项目中，多人组成的测试团队常具有下面这些角色：

(1) 测试经理：他们负责测试计划和测试统筹安排，具备软件测试、质量管理、项目管理和人员管理等领域的知识和经验，能指导和管理其他测试人员的工作。

(2) 测试设计人员：测试设计人员也可能被称为测试工程师、测试分析员、测试技术分析员等。他们需要掌握测试方法、流程和测试规格说明等，具备测试设计、测试分析以及软件工程等领域的知识和经验。

(3) 测试自动化人员：测试自动化人员不但具备测试的基础知识，还有编程经验以及丰富的测试工具和脚本语言知识。他们利用项目中提供的测试工具，按需要进行测试的自动化。经验丰富的还可以自己研发和定制项目特别需要的自动化测试工具。

(4) 测试环境管理员(实验室管理员)：负责测试环境的技术人员。一般是安装和操作测试环境方面的专家，具备系统管理员知识。建立、维护和支持测试环境，需要经常与系统管理员和网络管理员进行协调。他们也帮助一般测试工程师和开发工程师搭建测试环境。

(5) 测试执行人员：他们执行测试并编写缺陷报告，具备 IT 基础知识、测试基础知识，能应用测试工具，熟悉被测试对象。

3.2.3 测试团队与开发团队的比例

最初的软件研发过程中没有专职的测试团队，随着时代的发展，逐渐产生了专职的软件测试团队，进而产生了测试团队规模如何确定的问题。测试团队的规模又包括两个方面，一方面是测试团队整体的规模，最极端的讨论就是是否需要专职测试团队；另一方面是测试团队内部不同角色的成员的比例。当然，随着敏捷开发模式和云计算等要求快速发布的软件研发，测试工作是否由专职测试人员负责，还是由研发人员自己负责取决于企业和项目团队本身。测试团队的规模和比例取决于其他多种因素，本节将会仔细展开，进行讨论。同一情景下，采取不同团队建设策略，测试团队的规模和比例也不尽相同。

1. 质量风险

决定开发测试比例的第一因素是公司当前阶段产品所能承担的质量风险。不同公司对质量风险的承担度是不一样的，同样是做笔记本电脑，惠普这样的国际一线厂商和兼容机厂商显然对质量风险的承受度是不一样的。其次，即使是同一家公司，对不同产品的要求也可能不同，受到不同产品的用户群和产品对公司的重要性等因素影响。同一家公司的同一个产品在不同阶段对质量的要求也可能不同，在初期，为了抢占市场，抢占先机，质量要求不高，随着用户群扩大，用户期望值提高，质量要求会逐步上升。

2. 测试意识

决定开发测试比例的第二因素取决于开发工程师的如下重要素质：测试意识。要想提高软件产品质量，首先应该提升的是开发工程师的测试意识和能力，软件产品从开发工程师手中交付到测试工程师的时候，开发工程师应该进行过足够强度的测试。在惠普这样的国际一流软件企业中，很多资深软件开发工程师也具备高度的测试意识和技能。

3. 发布流程

影响开发测试比例的第三个因素是发布流程。一次性发布具有重大风险，而且犯错的代价比较昂贵，所以不得不投入更多的测试人员。行之有效的解决办法是分步发布。先内部用户试用，再请一部分外部用户试用。根据反馈，修复缺陷，最后发布给所有用户。即使出现问题，代价也相对较小。分步发布在当前软件业的一种形式是“公测”，请真实用户协助测试，得到反馈和确认后再进一步发布，如此反复迭代。在迭代中不断提高软件质量。根据近年来互联网软件产品分步式发布的经验，一方面可以满足用户急于看到新版本的渴望，另一方面风险可以极大得到控制，减少测试人员的投入。

4. 测试效率

影响开发测试比例的最后一个因素是测试效率。提升自动化率，引入持续集成等都是成功实践。自动化测试的最大应用领域是回归测试，如果实现了比较稳定的自动化测试，可以大大节约人力投入，加快软件发布进程。

5. 合理估计项目的开发测试比例的方法

(1) 首先看项目的性质，遇到问题影响范围是100%的核心业务，应该投入最多的测试人员，开发与测试之比至少为 1∶1，甚至 1∶2、1∶3 等。在航空航天软件领域，有的项目组，这一比例达到 1∶10 这个级别。

(2) 那些遇到缺陷影响范围可控或有替代方式的业务，上线步骤是递进的，开发和测试之比可以为 2∶1 或更高。

(3) 有些项目对质量要求不是很高，只需要做简单验证性测试即可发布，可以只设立一到两名测试工程师。

世界上没有万能钥匙，没有哪个开发测试比可以用在所有项目或产品中，最佳的比例一定是在实际工作中逐渐摸索出来的。

6. 手工测试工程师和自动化测试工程师的比例

这个问题涉及手工测试和自动化测试工作量的比例问题。一般来说自动化测试被认为适合在以下场景中使用：

(1) 重复性测试工作，如回归测试，尤其是回归测试用例数量多、人力不及的情景；

(2) 需要精确数据的场景，比如性能测试；

(3) 压力测试；

(4) API 级测试。

相比之下，手工测试更加灵活，具有更好的探索性，可以发现更多的缺陷，适应需求变化快速的项目。自动化测试的好处显而易见，难度也相对较大。所以出现了两种极端情况，一种是完全依靠手工测试，另一种是完全依靠自动化测试。在大部分完整的项目中，这两种极端情况都不可取。

影响自动化测试和手工测试的比例的因素有很多，影响比较大的有对项目质量的要求、项目使用的管理方法、实现测试自动化的技术难度、测试团队预算、测试工程师的水平等。

一般来说，项目质量要求越高，就需要更多的 API 级测试、“白盒”测试(比如代码覆盖)和其他自动化测试；管理方法越严谨，回归测试的测试用例越多，执行频度越高，自动化测试的工作也越多，相应的自动化测试工程师的比例也越大。随着基于互联网发布软件的模式日渐流行，一些软件产品相对简单、对质量的要求也不是特别高。在这种情况下，自动化测试工程师的比例就相对较小，甚至可以只配置手工测试工程师。

实现自动化的难易程度对于测试团队内部自动化测试工程师和手工测试工程师比例也有影响。一般来说，API 级的测试用例，比较容易实现自动化。图形界面的测试需要使用专门的工具，比如惠普的 UFT(前身为 QTP)自动化测试工具。UFT 对于 Windows 标准控件的支持就非常完善，可以精确地对 Windows 标准控件进行识别、操作和验证，实现测试用例的录制和回放工作。但是，包括 UFT 在内的绝大部分图形界面自动化测试工具都难以识别第三方控件，除非控件提供商在研发过程中加入特别的支持，比如实现了 MSAA 接口。另外针对视频、音频的测试也难以实现自动化。故而在决定测试团队里自动化测试工程师比例的时候，也要考虑到实现自动化的难度，有些测试工作虽然难以实现自动化，但可以依靠手工测试顺利完成。

根据以上讨论，测试团队中自动化测试工程师的比例受很多因素制约，在现实工作中有不同成功团队建设方案都可以参考，无论 2 比 8，还是 5 比 5，或是 1 比 20，可能都是合理的。实际工作中不一定非要削足适履般照搬别人的经验，要根据实际情况逐渐摸索确定合适的比例，找到最佳实践的人员配备比例。

3.3　测试团队的员工管理

在现今信息和高科技飞速发展的时代，软件开发项目必须有测试人员的参与，因而测试人才的培养和管理是非常重要的一个环节。打造高效的测试团队也越来越多地得到认可和重视。

3.3.1　测试工程师的职责

优秀的测试团队需要对不同的角色明确定义不同的角色职责。明确的职责是打造高效测试团队的前提。不同的开发项目和公司、团队会有各自适合的职责解释。

1. 按照职称划分

1) 测试经理

①编写或评审组织内部的测试方针；②选择合适的测试策略和方法，并引入或改善与测试相关的过程(缺陷管理、风险管理、配置管理等)，以便能够管理和控制变更，以及重现缺陷和测试活动；③作为整个项目的测试方面的代表与项目经理以及其他人共同制定测试计划并获取测试资源；④发起和监测测试工作，包括所有测试级别的测试设计、实现和执行。根据测试结果和测试进度(例如状态报告中的记录)及时调整测试计划；⑤对测试件进行配置管理，保证测试件的可追溯性；⑥利用合适的度量标准估算测试进度，评估测试和产品的质量；⑦选择和引入合适的测试工具，并为测试人员组织必要的培训；⑧确定测试环境和测试自动化的类型和范围；⑨根据测试过程中收集的信息编写测试报告。

2) 测试工程师

测试工程师在不同的公司可能有不同称呼。比如叫测试设计人员、测试分析员、测试技术分析员等。测试工程师的职责有：①分析、评审和评估用户需求、规格说明、设计和模型等内容的可测试性，并设计测试用例；②创建测试用例；③选择自动化测试用例；④准备和获取测试数据。

3) 测试自动化人员

测试自动化人员的职责有：①负责自动化测试框架的设计和搭建；②负责搭建自动化测试环境；③自动化测试需求分析，制定自动化测试计划；④开发自动化测试用例；⑤对自动化测试的执行情况进行分析，完善自动化测试框架。

4) 测试环境管理员

测试环境管理员的职责有：①负责测试环境所需的网络规划和建设，维护网络的正常运行；②建立、设置和维护测试环境所需的应用服务器或软件平台；③对实验室的硬件、软件资源进行登记、分配和管理；④申请所需的硬件和软件资源，协助有关部门进行采购和验收；⑤对使用实验室的硬件、软件资源的使用权限进行设计、设置，保证其安全保密性；⑥安装新的测试平台和被测试系统；⑦优化测试环境，提高测试环境中网络、服务器和其他设备的运行性能。

5) 测试执行人员

测试执行常常由测试工程师完成。他们按照预先已知的软件待测功能承担完成测试任务。测试执行人员的职责有：①分析、评审和评估用户需求、规格说明及可测试性，评审测试规格说明；②建立测试环境(通常需要与测试环境管理员相互配合，共同完成)；③执行各种级别的测试，记录测试日志，评估测试结果，记录实际结果和期望结果之间的偏差；④根据要求使用测试管理工具和测试监控工具；⑤实施自动化测试(可能需要开发人员和测试自动化人员的支持)。

根据项目或产品的测试级别及可能存在的测试风险，需要由不同的人充当不同的测试角色，同时保持一定的独立性。不同的测试级别，测试人员的组成可能是不一样的。例如，在组件和集成测试级别中，测试人员可能是开发人员；进行验收测试的测试人员一般是业务方面的专家或用户；而进行操作性验收测试的一般是将来使用软件的操作者。

2. 按照测试的工作方式划分

按照使用的测试方法可以将测试工程师大致分为两大类：一类以手工方式为主要工作模式，另一类以编程为主要工作方式。健康的软件产品或软件项目，对这两类测试工程师都需要。这两类工程师有各自的侧重点，职业发展道路也不尽相同。由于各种主客观原因，有部分测试工程师在手工测试工程师和自动化测试工程师这两个角色间相互转换。

1) 手工测试工程师

这也是较早期的软件测试工程师的形态，这些工程师以手工执行测试用例为主要方式进行工作。手工测试工程师手动执行测试用例，然后观察结果，和预期测试结果进行比较，并

判断测试用例是否通过，然后书写测试报告，对于失败的测试用例在软件缺陷跟踪管理系统中记录并跟踪。这个职位对计算机专业技能的要求不是很高，不要求很强的编程能力，甚至可以不具备编程能力，但是需要对用户需求有深刻了解，可以高度模拟用户行为，更需要严谨仔细的工作作风。部分情况下手工测试工程师需要执行自动化测试工程师开发的自动化测试用例。初级软件测试工程师通常从手工测试工作入手，在工作中不断提高自己的技术和相关业务技能，逐渐在技术这条职业发展路线上向前发展。

2) 自动化测试工程师

自动化测试是当代软件测试的发展趋势，在软件测试活动中，自动化测试的比例越来越高，对自动化测试工程师的需求也越来越多。自动化工程师通常具备不逊于开发工程师的编程能力，使用编程的方法开发自动化测试用例、自动化测试工具甚至自动化测试框架，为方便读者阅读，姑且把这些称为自动化测试工具。这些自动化测试工具通常适用于重复性测试活动(比如回归测试)和人工难以执行或难以精确执行的场景(比如性能测试和压力测试等)。自动化测试工具一方面极大提高了测试活动的效率，另一方面避免了手工测试引入的不稳定性。但是自动化测试不能解决所有问题，另外也引入了一些新问题，这些都是自动化测试工程师要面对和解决的问题。

3. 按照测试工作的内容划分

软件质量的不同侧面有不同的要求，于是测试工程师的工作内容就有不同的侧重点。尤其是大型软件系统，按照测试工作的内容划分得更细，术业有专攻，不同的工程师专注于软件质量的某个侧面。常见的角色划分有：功能测试工程师、性能测试工程师、安全测试工程师、国际化/本地化测试工程师、兼容性测试工程师以及用户体验工程师等。

1) 功能测试工程师

功能测试工程师是最常见的职位，主要职责是针对软件系统的功能进行测试，验证软件系统的功能符合系统设计，满足用户需求，尽量发现功能性软件缺陷并验证已经修复的功能性软件缺陷。功能测试工程师应深入理解软件系统的用户需求，能够模拟最终用户的行为，站在用户的角度对软件系统设计测试用例并执行测试。

一般认为功能测试工程师主要使用黑盒测试技术设计并执行测试用例，只需考虑各个功能，不需要考虑整个软件的内部结构及代码。黑盒测试的主要技术包括等价类划分、边界值分析、错误推测、因果图和基于模型的测试等方法。随着软件测试行业的发展，具备编程能力的测试工程师越来越多，这些工程师也使用白盒测试的方法设计、执行和改进功能测试。典型的白盒测试方法有代码审查、代码覆盖和错误注入等。这些黑盒和白盒测试方法在本丛书中有详细介绍，供读者参考学习。

2) 性能测试工程师

性能测试工程师的主要职责是对软件系统进行性能测试，根据软件系统的性能需求和用户对性能的要求设计具体的性能测试场景并执行性能测试用例。狭义上讲，软件性能衡量的是软件具有的响应时间能力；广义上讲，是软件对时间、CPU、内存和硬盘等资源的消耗进

行度量。性能测试工程师的要求比较高，一方面要设计典型性能测试场景和测试用例，保证性能测试的有效性，通常这需要对系统需求和用户行为有深刻的理解；另一方面，又要深入掌握计算机相关知识，具备较强编程能力，因为要获取精确的响应时间、CPU 时钟、内存消耗(系统、某进程或某线程消耗的内存)，就必须精通相关知识并通过编写程序的方法才可以实现。

3) 安全测试工程师

安全测试工程师负责在软件产品的生命周期中，对产品进行检验以验证产品符合安全需求定义和产品质量标准。提升软件系统的安全质量，在软件发布前找到安全缺陷并予以修复，验证软件系统内的保护机制能否应对非法入侵，在遇到非法入侵时，对其他子系统行为的影响。随着互联网软件在线发布形式的普及，对如今当红的社交网络、电子商务等包含敏感信息的大型软件系统，安全测试变得越来越重要。

安全测试涉及的方面很广，由于攻击者可以使用各种方法对软件系统进行攻击，故对于安全测试工程师来说，没有灵丹妙药，只能主动学习研究新的攻击方法，主动防御。但是对于已知的攻击方法和安全弱点：一方面，根据以往的经验，对系统使用已知的攻击方法进行针对性测试；另一方面，惠普这样的国际大型软件企业也有专业的安全测试工具，比如 HP WebInspect，安全测试工程师应熟练掌握这些工具的使用方法。

3.3.2 测试人员的综合技能

适合测试人员的综合技能包括：

(1) 问题的分析和解决能力：此能力对测试人员非常重要，因为对问题进行解剖和找出问题的症结是提高产品质量的关键。

(2) 精湛的技术：我们注重的是应聘者是否通晓网络和操作系统，不仅能写代码，而且能够优化代码。

(3) 项目管理：对测试人员来说，此能力是指如何有效支配个人的时间，以及如何策划和确保有许多互相牵制成分的计划按时完成。

(4) 对高质量有执着追求：如果不具备这个素质，应聘者就无法胜任任何工程技术工作，更不必说测试工作了。

(5) 自信：在日常工作中，测试人员找出的软件错误并不一定都能得到修正，在必要时，测试人员需要有自信而去据理力争。

(6) 冲击力和影响力：影响力来自于自信和经验，冲击力来自于敢于革新。多数应聘者在谈到如何给自己的公司带来变革，或者如何在学校带领团队出色地完成项目时，都会体现这个特征。

(7) 跨界合作：创新往往来自于各部门之间的合作，只顾埋头自己的项目，甘做井底之蛙的员工是不会成功的。

(8) 人际关系意识：主要指自我意识，许多优秀的应聘候选人能够认识到自己的不足之处，并且知道如何不断地提高自己，也就是有不断提升自身素质的计划。

(9) 相关业务知识。绝大部分软件系统都有其所服务的专业领域，比如电子商务、制造

业、物流、金融银行业等。作为测试工程师，首先要对软件系统所服务领域的专业知识有广泛、深入了解，这样才能更好地理解用户需求，模拟用户的心理和行为，进而设计出更符合实际的测试用例。就像很多企业或组织宣传的那样，以用户为本、为用户服务，了解用户、了解用户的处境以及他们的行为。

(10) 软件测试专业技能。软件测试行业发展到今日，已经有了比较成熟完善的专业知识和专业技能体系。从技术的角度有等价类划分、边界值分析、基于风险的测试、基于模型的测试等方法；从管理的角度，有回归测试、探索性测试等方法。作为专业软件测试工程师，必须熟练掌握并能在实际工作中灵活运用这些知识。

(11) 计算机体系结构。软件测试工程师也是计算机专业人员，想要更好地完成本职工作，必须对计算机体系结构有深刻了解。当遇到技术困难时，可独立解决这些技术难题，为顺利完成工作铺平道路。当遇到软件缺陷时，可以深刻洞悉缺陷的根本原因，有利于简化复现步骤，在报告中精确分析软件缺陷的原因，加快缺陷修复的过程，提高软件研发的效率。

(12) 编程技能。作为自动化测试工程师，应具备不弱于开发工程师的编程技能，才能够开发自动化测试用例和自动化测试工具。作为手工测试工程师，如果掌握一定的编程技能，尤其是脚本语言，可以将一部分简单任务自动化，大大提高工作效率，降低工作强度。

3.3.3　测试人员的职业发展

首先介绍一下广义的职业规划。职业规划就是：从事什么样的行业，在什么样的组织，担任什么样的职位，希望达成什么样的成就，如何通过学习与工作达到目标？做职业规划的意义在于帮助个人准确定位职业方向；评估个人目标和现状的差距；提供前进的动力和可行的方法；根据个人特点和强项，在职业竞争中找到合适的位置，发挥个人优势；全面了解自己，了解行业发展，与时俱进，增强职业竞争力；通过自我评估，知道自己的优缺点，通过反思和学习，不断完善自己，提高个人价值。职业规划的几个基本点：

- What you are? 自己是什么样的人？
- What you want? 想得到什么？
- What you can do? 能做什么？自己专业技能何在？
- What can support you? 你具有哪些职业竞争能力？
- What fits you most? 什么是最适合你的？
- What you can choose in the end? 现实条件下，可选择的余地有哪些？

根据以上讨论，具体落实到软件测试的职业发展规划中。我们可以建议职业规划的基本原则：择己所爱、择己所长、择世所需、择己所利。

我们需要了解测试的不同工作职位所需具备的技能是什么；自己目前是否已经具备胜任这个职位所需具备的技能；哪些方面已经胜任；哪些是短板；这些很难做到，在惠普等国际一流软件企业，通常会给新进员工指定一名有经验的员工做导师，对新晋员工的工作进行评价和指导，帮助其快速掌握技能和能够清晰认识自己当前的状态；如果没有导师，完全依靠自己领悟学习并在工作中进行实践，成长的道路会更加漫长而艰辛。

软件技术的快速发展和结合，迫使员工掌握的知识要不断更新，对测试人员也不例外。

要发挥人才优势和使就业的员工有发展空间，绝大多数高科技公司都有专门的职业发展规划和渠道。希望每一位员工都能热爱自己的本职工作，使各自的技术水平和其他能力达到相应技术级别和职位的要求，能在工作岗位上按照个人的意愿不断进步。这些公司往往给员工(包括项目管理人员、开发人员和测试人员)指出两条可以选择的职业生涯发展轨道："技术轨道"或"管理轨道"。

技术轨道指的是纯技术发展方向。成为一个或多个技术领域的专家：资深开发工程师、资深测试工程师等就是这样的例子。除非技术人员自己提出要求转变职业方向，公司招聘进来的技术人员自动地在技术轨道上发展。但角色转换也是可能的。比如测试人员去做开发人员，开发人员转成项目经理等。

1. 测试技术轨道

选择测试技术职业发展轨道的测试人员一般会经历如下阶段：

1) 初级测试工程师

要求具备必要的计算机知识和技能；掌握测试技能及方法，具有测试实施/执行能力。多数新招聘来的测试人员都处在这个级别。

2) 中级测试工程师

中级测试工程师要在初级测试工程师岗位上有两年以上工作经验。能胜任已分配的本职工作。具有测试设计能力，能够指导初级测试工程师工作。

3) 高级测试工程师

高级测试工程师必须在中级测试工程师岗位上有至少两年以上工作经验。一般要 5 年以上工作经验。他们具有测试规划及管理能力、测试分析及报告能力、测试过程设计及改进能力。可以指导中级测试工程师工作。

在技术这条发展道路上除了掌握对计算机知识和软件测试理论及技术的理解与应用水平外，其实对软件系统所应用的业务知识和使用人员的行为的了解也是一个非常重要的方面。比如为大型企业做 ERP 系统(Enterprise Resource Planning，企业资源计划系统)测试，就需要对企业的各种信息、资源及配备，尤其是供应链有深刻了解，深入了解不同用户需要什么样的功能与信息，如何运用这些信息做决策，这样在测试过程中才能真正判断软件系统是否满足用户的需求。再比如从事金融行业软件测试的测试工程师，一定要对金融知识有足够的了解，甚至有些从事金融软件测试的企业或部门鼓励软件测试工程师也参加 CFA(Chartered Financial Analyst，特许金融分析师)认证。

选择技术型职业发展路线的工程师再向上发展还有测试架构师这样的职位。

4) 软件测试架构师

软件测试架构师的工作包括：开发和设计测试框架测试库；纵横全局考虑产品的功能，设计复杂的测试系统；负责研发某一项特定的测试技术；为公司考虑如何提高测试效率。总的来说，我们可以这样描述：软件测试架构师领导公司测试技术的发展和测试策略的方向。

区别软件测试架构师和普通软件测试工程师的特质是：普通软件测试工程师关注的是某个功能模块、一条产品线，而软件测试架构师关注的是整个公司的测试部门的问题。甚至对一些更加资深的软件测试架构师，他们已经不再局限于产品当前版本的测试，他们可以前瞻性地考虑未来版本的测试策略和技术。测试架构师具备测试技术以及测试方法学雄厚的知识，不仅仅是公司内部的知识，也包括公司外部的知识。所以他们具备实力给那些测试经理提供“咨询”服务，告诉他们什么样的测试技术、什么样的测试平台符合公司要测试的产品，什么样的软件流程可以更好保证软件质量。

2. 测试管理轨道

测试管理是测试工程师的另一条重要的职业发展路线。测试管理者可以在测试员工梯队里向高层发展，管理更大规模的团队和领域，负责更大的产品或项目。管理型职业发展路线向上的方向是不断收缩的金字塔，能够达到类似技术副总裁这样的高级管理职位的测试管理者凤毛麟角。在管理型职业发展路线上通常设置测试主管、测试经理和测试总监这样的职位。

1) 测试主管

测试主管是测试管理的第一级，测试主管直接管理测试工程师，向测试经理汇报。一名测试主管通常管理由 2 到 10 人组成的测试团队。团队的规模大小取决于发布某一特定产品功能、组件(在云计算中动态分配 CPU 资源给需要的用户，在订票系统中用户订票事务过程)或共享的产品功能的工作量。这样的测试团队也可能负责某个特定的测试领域，比如性能、规模或安全性。所有这些职员由测试主管直接领导。在测试主管和一线测试工程师之间没有其他管理层。

2) 测试经理

测试经理是二线测试管理者，较少亲自做具体的测试工作，比如编写和执行测试用例。但每一个在测试领域的人，无论在什么级别，都会亲自动手寻找软件的缺陷。测试经理仍然需要懂得技术，但会被要求多注重于建立适合本部门的测试流程和工具，而不是在具体的功能测试上。测试经理的管理工作更多在于测试设备、人员等资源的分配与管理，不同项目间的合作与协调等方面。测试经理通常涉及最复杂的产品功能，或处理最复杂的协调工作。无论是否在管理职位上任职，每一个工程师都应该具备较强的动手能力和较高的技术水平。这种期望是以技术为导向企业文化的要求。

习题与思考题

1. 在测试团队管理中，软件测试组织有哪些角色？职责分别是什么？
2. 测试工程师、测试自动化人员和测试执行人员的职责是什么？有何区别？
3. 手工测试工程师和自动化测试工程师的工作内容分别是什么？
4. 比较功能测试工程师、性能测试工程师和安全测试工程师的工作内容？
5. 测试团队常见的交流沟通方法有哪些？

第4章　软件测试文档管理

测试是软件生命周期中一个独立关键的环节，也是保证软件质量的重要手段。为了使测试能够有计划、有条不紊、系统性地执行，就必须编制测试文档。测试文档的集合记录和描述了需要测试的目的、流程、范围、标准和缺陷信息等，测试文档本身同时也是软件测试工作的交付物之一。测试过程实施所必备的核心测试文档包括：测试计划、测试规范、测试用例和软件测试报告。本章将详细介绍软件测试文档的种类、内容、管理方法和业界的最佳实践。

4.1　测试文档的必要性和重要性

软件测试是一个复杂的过程，与软件研发过程中其他工作联系紧密，因此必须把对测试的要求、范围、过程及测试结果以正式的文档形式写下来。规范化的测试文档可以清晰地描述要执行的测试、策略、计划、规范及测试的结果，有助于包括软件测试团队、开发团队和项目经理在内的软件研发团队对测试工作有统一的认识，更好地对测试工作进行管理和监督。

4.1.1　测试文档的必要性

随着测试走向规范化管理，一套测试文档已成为测试团队必须完成的重要任务之一。软件测试是有计划、有组织和有系统的软件质量保证活动，而不是随意地、松散地、杂乱地实施过程。为了规范软件测试内容、方法和过程，在对软件进行测试之前，必须创建测试计划。

测试文档是一种工具，一个软件项目的测试是否高质量完成，一般可从两个方面进行评价：一是能否提供高质量的测试活动和结果，二是能否提供有效的测试文档。而对于后者，高质量的测试文档是体现前者是否高质量完成的证明。编制测试文档的必要性体现在以下几方面。

1. 提高项目测试过程的透明度

标准规范、齐全的文档会体现和记录测试过程中计划发生和已经发生的事件，便于测试人员掌握测试进度、测试质量以及各种测试资源的调配。同时，测试文档有助于测试人员与开发人员明确了解各自的职责，互通信息，共同把握测试和开发的进度和范围。

2. 文档化能规范测试，能提高测试效率

测试人员用一定的时间编制、整理测试文档，可使测试人员对各个阶段的工作都进行周密思考、找出存在的问题，从而减少差错，提高项目测试质量。例如：测试过程中肯定会遇到各种问题，诸如软件问题或测试设置等需要向开发组反馈来寻求解决，通过对文档的检查，

在项目测试早期发现文档错误和不一致之处，及时加以纠正，可以避免随着项目的深入而导致大问题的出现和为纠正失误而付出更大成本。

3. 便于团队成员之间的交流与合作

描述清楚、完备的测试文档便于项目组领导了解测试过程中的各项指标，在开发团队与测试团队之间架起一座桥梁。文档是“无声的语言”，它记录了项目测试过程中有关测试配置、测试运行、测试结果等方面的信息，有利于项目管理人员、测试人员之间的交流和合作。

4. 测试文档的重要性还表现在对于项目“传承”的重要性

有了好的文档，那么当项目有新成员进入，测试文档就可以承担起指导新成员快速工作的作用，而不是单单询问原来的成员，节省了大家的时间。当测试完毕后，测试文档将成为项目测试的文字载体，在后续人员培训方面提供详尽的素材。

5. 测试文档是测试人员经验提升的最佳途径

善于学习是不断进步所必需的动力和源泉。对于软件测试来说，项目测试文档对于项目测试人员的素质提升是很有效的。有些企业在进行项目测试时可能会出现一个通病：由于人员素质有限，许多决定只凭口头叙述，缺少足够的文字记录，以至出现问题时往往显得无所适从。从本质上讲，测试文档强调的是一种规范化管理，要求项目人员利用书面语言进行沟通表达，以指引项目运作。

测试人员不应该只为写测试文档而写文档，良好的文档是思想交流、沟通的基础，也是整理和理清思路的基础。不懂得从经验中学习和成长，永远不会有质的提高。每次完成一个测试任务，都有目的去总结，找到自己的不足之处，这样一个合格的测试人员才可能成长起来。

6. 有利于项目测试的监控作用

测试本身是一项风险很高的工程，需要进行严谨的项目监控。阶段性的检查、评审和文档化成果是重要的方法之一，详尽而规范的测试文档成果不仅有利于监控项目进度，也利于项目验收。

4.1.2　测试文档的重要性

项目测试文档是用来记录、描述、展示测试过程中一系列测试信息的处理过程，通过书面或图示的形式对项目测试活动过程或结果进行描述、定义及报告。例如，分阶段测试计划文档，测试流程文档，测试数据文档，测试参数设置文档和测试指南文档等。这些文档将伴随着软件测试的各个阶段逐渐充实、完善，同时也记载了整个测试的过程和成果。

以测试计划为例，软件测试计划作为软件项目计划的子计划，在项目启动初期是必须规划的。在越来越多公司的软件开发中，软件质量日益受到重视，测试过程也从一个相对独立的步骤变得越来越紧密嵌套在软件整个生命周期中。如何规划整个项目周期的测试工作；如何将测试工作上升到测试管理的高度都依赖于测试计划的制定。测试计划因此也成为开展测试工作的基础。

编写软件测试计划的重要性在于它使整个测试团队的成员拥有了测试规范指南，能够了

解各自负责测试的组件、功能和其他分配的测试任务，并减少重复和不清晰的任务分配。同时测试文档应起的作用还包括帮助测试人员在测试执行过程中发现更多的软件缺陷。因此，软件测试计划中的测试范围必须高度覆盖功能需求，测试方法必须切实可行，测试工具具有较高的实用性，便于使用，生成的测试结果直观、准确。

一个好的测试计划可以起到如下重要作用：

(1) 避免盲目的、无从检查的测试执行；

(2) 使测试工作和整个开发工作融合起来；

(3) 把有限资源和变更事先作为一个可控制的风险。

4.2　测试文档规范

软件测试是一个复杂过程，与软件研发过程中其他工作联系紧密，因此必须把对测试的要求、范围、过程及测试结果以正式的文档形式写下来。规范化的测试文档可清晰地描述要执行的测试、策略、计划、规范及测试的结果，有助于包括软件测试团队、开发团队和项目经理在内的软件研发团队对测试工作有统一的认识，更好地对测试工作进行管理和监督。为使测试文档的制定和编写更符合国际和国内业界权威标准，我们将重点介绍两个相应的测试文档编制参考标准。

国家标准《计算机软件文件编制规范》

为规范软件测试文档的管理，中华人民共和国国家质量监督检验检疫总局中国国家标准化管理委员会发布了 GBT9386-2008《计算机软件测试文档编制规范》。该标准是对 GB/T 9386《计算机软件测试文件编制规范》的修订。

该标准规定了各个测试文档的格式和内容，涉及测试计划、测试说明和测试报告等。

1. 测试计划

描述测试活动的范围、方法、资源和进度。它规定被测试的项、被测试的特征、应完成的测试任务、负责每项工作的人员以及与本计划有关的风险等。

2. 测试说明

包括三类文档。

1) 测试设计说明：详细描述测试方法，并标识该测试设计和相关测试所覆盖的特征，还标识为完成测试和规定特征的通过准则所需要的测试用例和测试规程；

2) 测试用例说明：将用于输入的实际值以及预期的输出形成文档。并标识在使用具体测试用例时对测试规程的约束。将测试用例与测试设计分开，可以使它们用于多个设计，并能在其他情形下重复使用；

3) 测试规程说明：标识为实施相关测试设计而运行系统并执行规定测试用例所要求的所有步骤。测试规程与测试设计分开，特意明确要遵循的步骤，而不宜包含无关细节。

3. 测试报告

包括 4 类文档：

1) 测试项传递报告：指明在开发组和测试组独立工作的情况下，或者在希望正式开始测试的情况下为进行测试而被传递的测试项；

2) 测试日志：测试组用于记录测试执行过程中发生的情况；

3) 测试事件报告：描述在测试执行期间发生并需要进一步调查的任何事件；

4) 测试总结报告：用来总结测试活动和结果的文档。

这些文档与其他文档在编制方面的关系以及与测试过程的对应关系如图 4-1 所示。

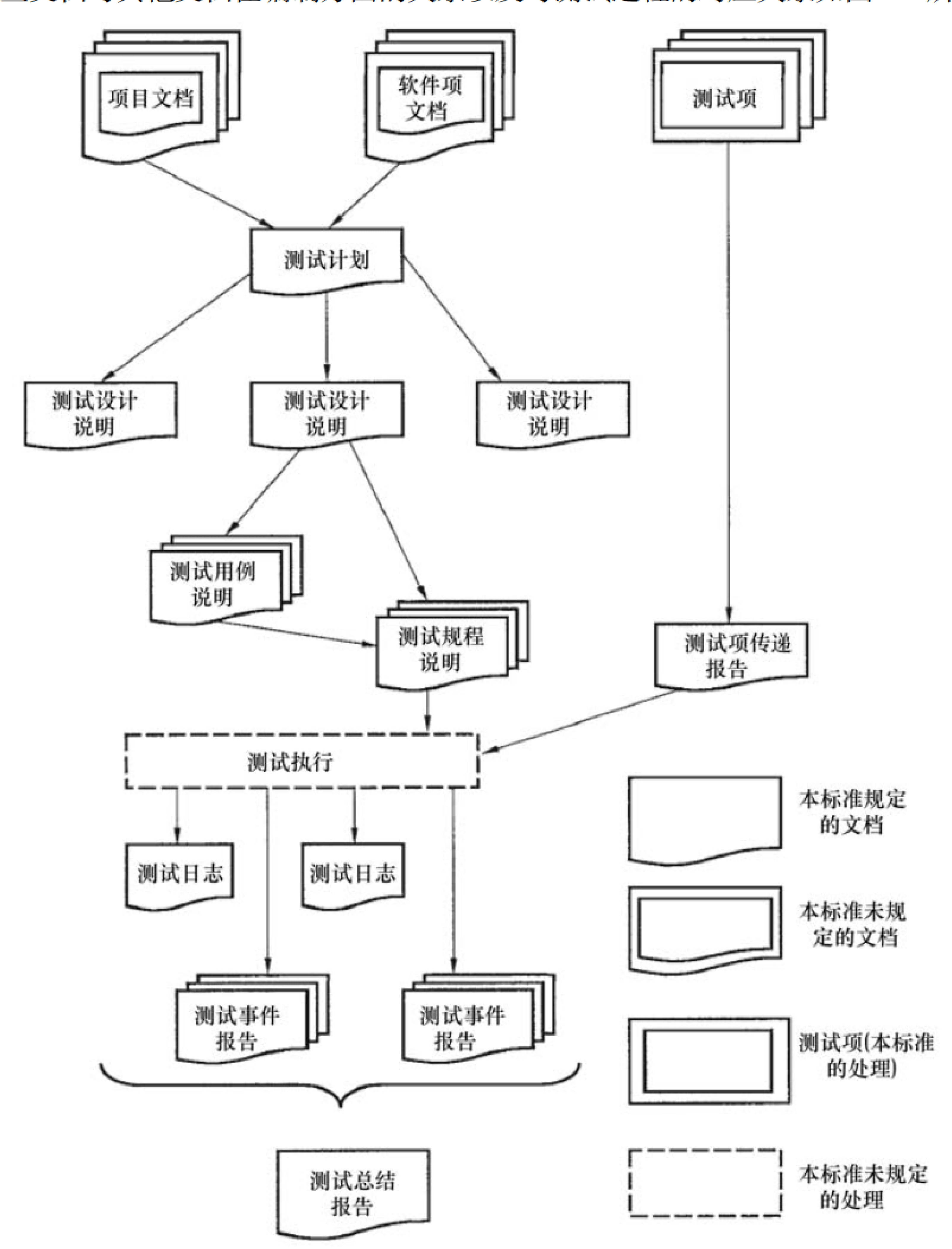


图 4-1　测试文档与其他文档关系图

4. 测试计划

在 GBT9386-2008 中，对测试计划的结构做了严格的描述，其结构如下：

1) 目的

用来描述测试活动范围、方法、资源和进度。定义被测试的软件项、要测试的特征，要完成的测试任务、负责每项任务的人员以及与该计划相关的风险。

2) 提纲

(1) 测试计划标识符；
(2) 引言；
(3) 测试项；
(4) 需要测试的特征；
(5) 不需要测试的特征；
(6) 方法；
(7) 测试项通过准则；
(8) 暂停准则和恢复要求；
(9) 测试交付项；
(10) 测试任务；
(11) 环境要求；
(12) 职责；
(13) 人员配备和培训要求；
(14) 进度；
(15) 风险和应急；
(16) 批准。

上述各项应按规定的顺序排列。附加项可以直接加在批准项之前。如果上述某一项的部分内容或全部内容是在另一个文档中，则可以列出引用材料的出处以替代相应的内容。引用的内容必须附在测试计划中或向该计划的用户提供。

3) 详细说明

(1) 测试计划标识符：为该测试计划指定一个唯一标识符。

(2) 引言：概述要测试的软件项和软件特征。可以包括每个测试项的要求及其历史记录。如果存在下述文档，在最高层测试计划中需要引用它们：项目授权；项目计划；质量保证计划；配置管理计划；有关的政策或方针；有关规定。

在多级测试计划中，每个低一级的测试计划必须引用上一级测试计划。

(3) 测试项：标识测试项(其中包括其版本/修订级别)，并在测试开始之前规定影响硬件需求的测试项传递媒体的特性，或者指出逻辑或物理变换要求(例如，程序必须从磁带转换到磁盘上)。如果存在需求规格说明、设计规格说明、用户指南、操作指南、安装指南等测试项文档集则需要引用之，引用与这些测试项有关的任何事件报告。可以标识那些明确排除在测试之外的软件项或测试项。

(4) 需要测试的特征：标识所有要测试的软件特征及其组合，并标识与每个特征或每个特征组合有关的测试设计说明。

(5) 不需要测试的特征：标识不要测试的所有特征和重要的特征组合及其理由。

(6) 方法：描述测试的总体方法。对于每个主要的特征组或特征组合组，规定要确保这些特征组得到充分测试的方法。规定用于测试指定特征组所需的主要活动、技术和工具。应详尽地描述方法，以便标识出主要的测试任务，并估计执行各项任务所需的时间。规定所希望的最低程度的测试充分性，指明用于判断测试充分性的技术(例如，确定哪些语句至少已执行过一次)。规定任何补充的结束准则(例如，出错频率)。应规定用来跟踪需求的技术。标识对测试的主要约束，例如：测试项的可用性、测试资源的可用性和测试截止期限等。

(7) 测试项通过准则：规定用来确定每个测试项是否通过测试或者测试失败的准则。

(8) 暂停准则和恢复要求：规定用于暂停与该计划有关的测试项的全部或部分测试活动的准则。规定恢复测试时必须重复的测试活动。

(9) 测试交付项：标识可交付的文档，建议包括以下文档：测试计划、测试设计说明、测试用例说明、测试规程说明、测试项传递报告、测试日志、测试事件报告、测试总结报告等。测试输入数据和测试输出数据应标识为可交付项。测试工具(例如：模块驱动器和桩模块)也可以包含在内。

(10) 测试任务：标识准备和执行测试所需的任务集合。标识各项任务间的所有依赖关系和所要求的任何特殊技能。

(11) 环境要求：详细说明测试环境必需的特性。详细内容应包括各种设施的物理特征。这些设施包括硬件、通信和系统软件、使用方式(可单独使用)以及支持测试所需的任何其他软件或设备。还应规定这些测试设施、系统软件和专有组成部分(例如：软件、数据和硬件)所需的安全等级。标识必要的特殊测试工具及其他任何测试要求(例如：出版物或办公场地等)。标识测试组目前尚不可用的所有需要的来源。

(12) 职责：标识负责管理、设计、准备、执行、监督、检查和解决的各个小组。另外，标识负责提供标识的测试项和标识的环境要求的各小组。这些小组可以包括开发人员、测试人员、操作员、用户代表、技术支持人员、数据管理员和质量保证人员。

(13) 人员配备和培训要求：按技能等级提出测试人员配备要求。标识为提供必要技能的培训选项。

(14) 进度：包括在软件项目进度中标识的测试里程碑以及所有测试项传递事件。定义所需的其他测试里程碑，估计完成每项测试任务所需的时间，为每项测试任务和测试里程碑规定进度，对每种测试资源(即设施、工具和人员)规定使用期限。

(15) 风险和应急：标识测试计划的高风险假设，对各种风险提出应急措施(例如：测试项的延期交付可能需要加班以满足交付日期)。

(16) 批准：确定必须批准该计划人员的姓名和职别。为签名和填写日期留出位置。

5. 测试设计说明

在 GBT9386-2008 中，对测试设计的结构做了严谨的描述，其结构如下。

1) 目的

通过测试设计及其相关测试来详细地规定测试方法和标识要测试的特征。

2) 提纲

测试设计说明应有如下结构：

(1) 测试设计说明标识符；

(2) 要测试的特征；

(3) 方法细化；

(4) 测试用例标识；

(5) 特征通过准则。

上述各项应按指定的顺序排列。附加的项可以放在结尾处。如果上述某一项的部分内容或全部内容是在另一个文档里，则可以列出引用材料的出处以代替相应的内容。引用的内容必须附在测试设计说明中或向该设计说明的用户提供。

3) 详细说明

(1) 测试设计说明标识符：为该测试设计说明指定唯一的标识符。如果在相关的测试计划中有规定，则应引用。

(2) 要测试的特征：标识测试项，并描述作为该设计说明对象的特征和特征组合。尽管可能还有某些其他特征，但不必标识它们。

(3) 方法细化：将测试计划中描述的方法进行细化，包括要采用的具体测试技术。应标识分析测试结果的方法(例如：比较程序或可视化审查)。指明为选择测试用例提供合理依据的任何分析结果。例如：人们可以规定容错的条件(例如：区别有效输入与无效输入的那些条件)。归纳任何测试用例的共同属性，可以包括各种输入约束(如针对一组相关测试用例的所有输入必须是真)、任何共享环境的要求、任何共享特殊规程的需求以及任何共享测试用例之间的依赖关系。

(4) 测试用例标识：列出与该设计有关的每一测试用例的标识并简要描述。某个特定的测试用例可能在两个以上的测试设计说明中出现。列出与该测试设计说明有关的每个规程的标识及其简要描述。

(5) 特性提供准则：给出用于判定特征或特征组合是否通过或失败的准则。

6. 测试用例说明

在 GBT9386-2008 中，对测试用例的结构做了严格的描述，其结构如下。

1) 目的

定义由测试设计说明所标识的测试用例。

2) 提纲

测试用例说明应有如下结构：

(1) 测试用例说明标识符；

(2) 测试项；

(3) 输入说明；

(4) 输出说明；

(5) 环境要求；

(6) 特殊的规程要求；

(7) 用例间的依赖关系。

上述各项应按指定的顺序排列，附加项可以放在结尾处。如果上述某一项的部分内容或全部内容是在另一个文档里，则可以列出引用材料的出处以替代相应的内容。引用的内容必须附在测试用例说明中或向该测试说明的用户提供。

鉴于测试用例可能被不同小组的若干测试设计说明长期引用，为此在测试用例说明中必须包含足够具体的信息以方便重复使用。

3) 详细说明

(1) 测试用例说明标识符：为该测试用例说明规定唯一的标识符。

(2) 测试项：标识并简要描述该测试用例要执行的软件项和特征。对于每一测试项，考虑引用以下测试项文档集：需求规格说明、设计规格说明、用户指南、操作指南、安装指南。

(3) 输入说明：规定执行测试用例所需的每种输入。有些输入可以用值(必要时允许适当的容差)来规定，而其他输入(如常数表或事务处理文件)可用名称来规定。规定所有合适的数据库、文件、终端消息、内存驻留区域及操作系统传送的各个值。规定输入之间的所有必要的关系(例如：定时)。

(4) 输出说明：规定测试项所有要求的输出和特征(例如：响应时间)。为每个要求的输出提供准确的值(必要时允许适当的容差)。

(5) 环境要求：硬件规定执行该测试用例所需的硬件特性和配置(例如：132 个字符×24 行的显示器)；软件规定执行该测试用例所需的系统软件和应用软件，可包括操作系统、编译程序、模拟程序和测试工具之类的系统软件。

(6) 特殊的规程要求：描述对执行该测试用例的测试规程的任何特殊约束。这些约束可以包括特殊的装配或设置、操作者的干预、输出确定规程以及特定的清除过程。

(7) 用例间的依赖关系：列出执行该测试用例之前必须执行的各个测试用例，并概要说明这些测试用例之间依赖关系的性质。

7. 测试规程说明

在 GBT9386-2008 中，对测试规程做了严谨的描述，其结构如下。

1) 目的

详细说明执行一组测试用例的各个步骤，或者更广泛地说明为了评估一组特征而用于分析软件项的各个步骤。

2) 提纲

测试规程说明应有如下结构：

(1) 测试规程说明标识符；

(2) 目的；

(3) 特殊要求；

(4) 规程步骤。

上述各项应按指定的顺序排列。如有必要，附加项可放在结尾处。如果上述某一项的部分内容或全部内容是在另一个文档中，则可以列出引用材料的出处以代替相应的内容。引用的内容必须附在测试规程说明中或向该规程说明的用户提供。

3) 详细说明

(1) 测试规程说明标识符：为该测试规程说明指定唯一的标识符，必要时提供对相关测试设计说明的引用。

(2) 目的：描述该规程的目的。如果该规程执行任何测试用例，则提供对每个测试用例说明的引用。另外，提供对测试项文档相关部分的引用(例如：对使用规程的引用)。

(3) 特殊要求：标识执行该规程所需要的任何特殊要求。这些要求可以包括必要的规程、专门技能要求和特殊环境要求。

(4) 规程步骤：如适用，应包括在下述步骤。

- 日志：描述用来记录测试的执行结果、观察到的事件以及与测试有关的任何其他事件的任何特殊的方法或格式。
- 准备：描述准备执行该规程所需的动作序列。
- 启动：描述开始执行该规程所需的动作。
- 处理：描述在该规程执行过程期间所需的动作。
- 测量：描述如何进行测试的测量(例如，描述如何利用网络模拟器来测量远程终端的响应时间)。
- 暂停：描述当发生意外事件而暂停测试所需的动作。
- 重启动：标识任何规程的重启动点，并描述在每个重启动点重新启动规程所需的动作。
- 停止：描述正常停止执行时所需的动作。
- 清除：描述恢复环境所需的动作。
- 应急：描述处理在执行过程中可能发生的异常事件所需的动作。

8. 测试日志

在 GBT9386-2008 中，对测试日志做了严谨的描述，其结构如下。

1) 目的

按时间顺序提供关于执行测试的相关细节的记录。

2) 提纲

测试项日志应有如下结构：

(1) 测试日志标识符；

(2) 描述；

(3) 活动和事件条目。

上述各项应按指定的顺序排列。附加项可放在结尾处。如果上述某一项的部分内容或全部内容是在另一个文档中，则可以列出引用材料的出处以代替相应的内容。引用的内容必须附在测试日志里或向记录的用户提供。

3) 详细说明

(1) 测试日志标识符：为该测试日志规定唯一的标识符。

(2) 描述：除了在日志条目中特别注明外，此处应包括适用于日志中所有条目的信息。以下信息应予考虑：

- 标识被测试的各个测试项及其版本/修订级别。对于其中的每一项，如果存在其传递报告，则应加以引用；
- 标识执行测试的环境属性，其中包括设施说明、使用的硬件(例如：使用的内存容量、CPU 型号、磁带机的型号和编号、海量存储设备)、使用的系统软件及可用资源(例如：可用的内存容量)。

(3) 活动和事件条目：对于每个事件(包括事件的开始和结束)，要记录发生的日期和时间以及记录者的身份。以下信息应予考虑：

- 执行描述：记录正在执行的测试规程的标识，提供并引用该测试规程说明。记录执行时在场的所有人员(包括测试者、操作员和观察员)，还要说明每个人的职责。
- 测试结果：对每次执行，记录目视可观察到的结果(例如：产生的出错消息、异常终止和对操作员动作的请求等)。还要记录任何输出的位置(如磁带编号)，以及记录测试的执行是否成功。
- 环境信息：记录与本条目有关的任何特殊环境条件(例如，硬件更换)。
- 异常事件：记录某个不期望事件(例如：尽管请求显示总计并显示了正确的屏幕，但响应时间似乎过长。重复执行时，响应时间也同样过长)发生前后的情况。
- 事件报告标识符：随时记录每个测试事件报告产生的标识符。

9. 测试事件报告

在 GBT9386-2008 中，对测试事件报告做了严谨的描述，其结构如下：

1) 目的

将测试过程中发生的需要调查研究的所有事件形成文档。

2) 提纲

测试事件报告应有如下结构：

(1) 测试事件报告标识符；

(2) 摘要；

(3) 事件描述；

(4) 影响。

上述各项应按指定的顺序排列。附加的各项可以放在结尾处。如果上述某一项的部分内容或全部内容是在另一个文档中，则可以列出引用材料的出处以代替相应的内容。引用的内容必须附在测试事件报告中或向事件报告的用户提供。

3) 详细说明

(1) 测试事件报告标识符：为该测试事件报告规定唯一的标识符。

(2) 摘要：概述事件。标识所涉及的所有测试项，指出其版本/修订级别。应提供对有关测试规程说明、测试用例说明和测试日志的引用。

(3) 事件描述：对事件进行描述。该描述应包括：输入、预期结果、实际结果、异常现象、日期和时间、规程步骤、环境、重复执行的意图、测试者、观察者等内容。该描述应包括可能有助于隔离并纠正事件原因的相关活动和观察结果(例如，描述可能对此事件有影响的所有测试用例执行情况，描述与已公布的测试规程之间的任何偏差)。

(4) 影响：在所了解的范围内指明该事件对测试计划、测试设计说明、测试规程说明或测试用例说明所产生的影响。

10. 测试总结报告

在 GBT9386-2008 中，对测试总结报告做了严谨的描述，其结构如下。

1) 目的

总结指定测试活动的结果并根据这些结果进行评价。

2) 提纲

测试总结报告应有如下结构：

(1) 测试总结报告标识符；

(2) 摘要；

(3) 差异；

(4) 测试充分性评价；

(5) 结果汇总；

(6) 评价；

(7) 活动总结；

(8) 批准。

上述各项应按指定的顺序排列。附加项可直接加在批准项之前。如果上述某一项的部分内容或全部内容是在另一个文档中，则可以列出引用材料的出处以替代相应的内容。引用的内容必须附在测试总结报告或向该总结报告的用户提供。

3) 详细说明

(1) 测试总结报告标识符：为该测试总结报告指定唯一的标识符。

(2) 摘要：总结对测试项的评价。标识已测试的各个项，指出其版本/修订级别，并指出执行测试活动所处的环境。对于每个测试项，如果存在测试计划、测试设计说明、测试规程说明、测试项传递报告、测试日志和测试事件报告文档，则应提供对相关信息的引用。

(3) 差异：报告测试项与其设计说明之间的任何差异，并指出与测试计划、测试设计或测试规程之间的任何差异，详细说明每种差异产生的原因。

(4) 测试充分性评价：如果有测试计划，应根据测试计划中规定的测试充分性准则对测

试过程做出评价。确定未做充分测试的特征或特征组合，并说明理由。

(5) 结果汇总：汇总测试的结果。标识已解决的所有事件，并总结其解决方案。指出尚未解决的所有事件。

(6) 评价：对每个测试项(包括其限制)进行总体评价。该评价必须以测试结果和测试项级别的通过准则作为依据。可以包含对失败风险的估计。

(7) 活动总结：总结主要的测试活动和事件。总结资源消耗数据，例如：人员的总体配备水平和每个主要测试活动所花费的时间。

(8) 批准：详细说明必须批准该报告的所有人员的姓名和职务，并为签名和日期留出位置。

4.3　常用测试文档

前文介绍了软件测试文档的中国国家标准和国际标准，在实际软件测试工作中，每个项目都有各自的特点，不应该生搬硬套地适应，而应该参考选用适用的，并根据需求增减和调整。本节根据业界通行做法对常用的 6 种测试文档的形式、内容和写作方法进行简要介绍。

4.3.1　测试策略

软件测试策略的定义是：在一定的软件测试标准、测试规范的指导下，依据测试项目的特定环境约束而规定的软件测试的原则、方式、方法的集合。通俗地讲，测试策略描述了要进行哪些种类的测试(功能测试、性能测试、安全性测试、兼容性测试、文档测试等)和如何测试(手工测试、自动化测试等)的问题。

软件测试策略随着软件生命周期的变化、软件测试方法、技术与工具的不同而发生变化。这就要求我们在制定测试策略时，应该综合考虑测试策略的影响因素及其依赖关系。这些影响因素可能包括：软件产品的质量要求、应用领域、测试项目资源因素、项目的约束和测试项目的特殊需要等。以下简述制定软件测试策略的过程。

首先要明确制定软件测试策略的输入。这些输入包括：需要的软硬件资源的详细说明；针对测试和进度约束而需要的人力资源的角色和职责；测试方法、测试标准和完成标准；目标系统的功能性和技术性需求；系统局限(即系统不能够提供的需求)等。

其次要明确软件测试策略的输出。其中包括：得到最终确认的测试策略文档、测试计划和测试用例；需要解决方案的测试项目。

明确了软件测试策略的输入和输出后，制定软件测试策略的过程如下：

1. 确定测试的需求

测试需求指出测试内容，即测试的具体对象。在分析测试需求时，可应用以下几条一般规则：

测试需求必须是可观测、可测评的行为。如果不能观测或测评测试需求，就无法对其进行评估，以确定需求是否已经满足。每个用例或系统的补充需求与测试需求之间不存在一对一的关系。用例通常具有多个测试需求；有些补充需求将派生一个或多个测试需求，而其他

补充需求(如市场需求或包装需求)将不派生任何测试需求。

测试需求可能有许多来源，其中包括用例模型、补充需求、设计需求、业务用例、与最终用户的访谈和软件构架文档等。应该对所有这些来源进行检查，以收集可用于确定测试需求的信息。

2. 评估风险并确定测试优先级

成功的测试需要在测试工作中成功地权衡资源约束和风险等因素。为此，应该确定测试工作的优先级，以便先测试最重要、最有意义或风险最高的用例或构件。为确定测试工作的优先级，需执行风险评估和实施概要，并将其作为确定测试优先级的基础。

制定测试策略不仅是测试团队的工作，更需要与项目中的其他同事一起，将关注点放在测试需要解决的问题上，制定一个长期的解决方案，可以在整个项目周期内实施。除了上面列出的那些问题外，项目解决方案还要满足测试策略的基本需求：在开发周期内，尽早找到最严重的缺陷，提高软件质量。想尽早地发现最严重的缺陷，需要把项目的测试部分和开发部分有机结合在一起，包括不同的测试阶段、测试类型、项目环境，以及如何在环境、角色、职责之间升级代码，还有普遍使用的工具。

初始的测试策略应该尽量简单，甚至可以直接将其记录在白板上。简洁到可以把它的含义解释给项目组内的任何一个人，并保证他们能够理解这些策略的正确含义。在把测试策略细化并把细节写入文档之前，清晰地定义简化的概念是简单性的保证。根据一些团队的实践，把测试策略草稿的设想和内容写到白板或者内部 Wiki 上以帮助人们共享意见和观点，有利于整个软件项目团队其他成员共同参与软件测试策略的制定。在使用白板或者内部 Wiki 时，使用简洁易懂的示意图和流程图有助于团队中其他成员理解测试策略的内容。

3. 确定测试策略

一个好的测试策略应该包括：实施的测试类型和测试的目标、实施测试的阶段、技术、用于评估测试结果和测试是否完成的评测和标准、对测试策略所述的测试工作存在影响的特殊事项等内容。

如何才能确定一个好的测试策略呢？可以从基于测试技术的测试策略、基于测试方案的测试策略两个方面来回答这个问题。

第一方面，基于测试技术的测试策略的要点：任何情况下都必须使用边界值测试方法；必要时使用等价类划分方法补充一定数量的测试用例；对照程序逻辑，检查已设计出的测试用例的逻辑覆盖程度，看是否达到了要求；如果程序功能规格说明中含有输入条件的组合情况，则可以选择因果图方法。

第二方面，基于测试方案的测试策略：对于基于测试方法的测试策略，一般来说应该考虑如下方面：根据程序的重要性和一旦发生故障将造成的损失来确定它的测试等级和测试重点；认真研究，使用尽可能少的测试用例发现尽可能多的程序错误，避免测试过度和测试不足。

软件测试的策略、方法和技术是多种多样的。对于软件测试技术，可以从不同的角度加以分类：从是否需要执行被测软件的角度，可分为静态测试和动态测试。从测试是否针对系统的内部结构和具体实现算法的角度来看，可分为白盒测试和黑盒测试。从测试切入系统的

层级看，可分为单元测试、模块测试、集成测试和系统测试。从测试的分类看，可以分为功能测试和非功能测试，如性能测试、安全测试、国际化本地化测试等。

4.3.2　测试计划

测试计划定义为："一个叙述了预定的测试活动的范围、途径、资源及进度安排的文档。它确认了测试项、被测特征、测试任务、人员安排以及任何偶发事件的风险。"。软件测试计划是指导测试过程的纲领性文件，是测试文档中的重中之重。它包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。借助软件测试计划，参与测试的项目成员，尤其是测试管理人员，可以明确测试任务和测试方法；保持测试实施过程的顺畅沟通，跟踪；把控测试进度；应对测试过程中的各种变更。

下面介绍典型的测试计划的结构和如何编写测试计划。

1. 测试计划的目标

软件测试是有计划、有组织和有系统的软件质量保证活动，而不是随意地、松散地、杂乱地实施过程。为规范软件测试内容、方法和过程，在对软件进行测试之前，必须创建测试计划。如果没有开发对程序的行为作出定义，测试人员就很难执行测试任务。同样，如果没有对测试资源配置，测试进度做一些统筹，整个项目也很难成功。因此，软件测试计划是软件测试人员与开发人员交流的主要手段。

2. 测试计划的内容

1) 测试对象

测试过程的第一个论题就是确定测试的对象。虽然这是项目组成员都必须清楚的内容，但常常被忽视。他们可能被认为"太过明显，不言自明"，每个人都应该了解，但是，作为测试人员，永远不要假定任何事。

在软件定义阶段产生的可行性报告、项目实施计划、软件需求说明书或系统功能说明书，在软件开发阶段产生的概要测试说明书、详细设计说明书以及源程序等都是软件测试的对象。

2) 测试内容

测试计划需要明确在项目中工作的人、人员在做什么以及怎样和他取得联系。在小项目中这似乎没有太大必要，但即使是小项目成员也可能分散在不同的地方或者人员会时常变动，这为跟踪"谁做什么"造成麻烦。大型项目可能有上百个联络点，测试小组很可能要和所有人打交道，知道他们是谁和如何联系非常重要。测试计划应该包括项目中所有主要人员的姓名、职务、地址、电话号码、电子邮件和责任范围等。

同样，相关文档存放在哪里、软件从哪里下载、测试工具从哪里得到等都需要明确。

如果在执行测试时，硬件不可缺少，那么它们放在哪里，如何得到？如果有进行配置测试的外部测试实验室，那么它们具体在哪里？他们的进度安排是怎样的？这也就是通常所说的"Ramp-Up"文档。

3) 术语定义

对于测试文档中的一些术语，可能存在误解。在不同的项目组合作时，一个容易忽视的问题是：程序员、测试人员和管理部门都有各自的理解，如果测试人员和程序员对缺陷定义没有达成一致，争执分歧就在所难免。软件测试计划过程就必须包含小组成员用词和术语的定义。务必要求同存异，使全体人员说法一致。

以下是一些常用的术语和定义。不要把它当做是一个完整的词汇列表或者明确定义。它取决于具体项目、开发小组遵循的开发模式，以及小组成员的经验。给出这个列表的目的就是要开拓思路，想想应该为项目定义哪些内容，需要让全体人员了解其含义：

(1) TRD(Test Realse Document，测试发布文档)：程序员发布的文档，其中每个构造文件都是声明新特性、修复问题和准备测试的内容。

(2) Alpha 版：意在对少数主要客户和市场进行数量有限的发布。用于演示目标的早期实现。在实际环境中使用。使用 Alpha 版的所有人员必须了解确切内容和质量等级。

(3) Beta 版：意在向潜在客户广泛的分发正式的实现。这个阶段的测试也成为 Beta 测试。

(4) Triage Meeting(三方会审)：由测试管理员、项目管理员、开发管理员和产品支持管理员组成的团队，每周召开会议审查软件 Bug，并确定哪些需要修复，应该如何修复。

4) 确定测试范围

对于一些软件产品，某些内容不必测试。这些内容可能是以前发布过或者测试过的软件部分或者来自第三方的内容可以直接接受的。

计划过程需要验明软件的每一部分，知道它是否要测试，如果没有测试，需要说明这样做的理由。如果由于误解使得部分代码在整个开发周期漏掉，未做任何测试，这将会是灾难性的后果。

5) 测试阶段

要计划测试阶段，取决于项目的开发模式。在边写边改的模式中，可能只有一个测试阶段——某个成员宣布自己的工作完成时进行测试。在流水线和螺旋模式中，从检查产品说明书到验收测试可能有几个阶段。测试计划属于其中一个测试阶段。

测试计划过程应该明确每一个预定的测试阶段，并且通知项目小组。该过程一般有助于整个小组形成和了解全部开发模式。

注意：与测试阶段相关联的两个重要原则是进入和退出规则。每个阶段都必须有客观定义的规则。例如：说明书审查阶段在正式说明书审查公布时要结束。假如没有明显的进入和退出规则，测试工作就会瓦解为毫无意义的单个工作。

6) 测试策略

测试策略描述测试小组用于测试整体和每个阶段的方法。回顾到目前为止的软件测试知识，如果面对需要测试的产品，需要决定使用“黑盒”测试还是“白盒”测试好。如果综合使用两种技术，那么在软件的哪些部分、什么时候去运用它？

某些功能需要手工测试，而其他可以用工具和自动化测试来完成。如果要使用工具，是否需要二次开发？或者能够买到已有的商用解决方案？也许更有效的方法是把整个测试工作提交给某个专业的测试公司，只要本方的测试人员监督他们的工作质量即可。

做决策是一项复杂工作，需要有经验相当丰富的测试人员来做，因为这将决定测试工作的成败。

7) 资源要求

计划资源要求是确定实现测试策略必备条件的过程。在项目期间测试可能用到的任何资源都要考虑到。例如：

(1) 人员：人数、经验、专业、全职、兼职、实习生。

(2) 设备：计算机、测试硬件、打印机、测试工具等。

(3) 办公及实验室空间：可容纳多少人、多少台机器。

(4) 软件：文字处理软件、数据库软件以及一些工具类软件。

(5) 外包测试公司：是否需要用测试提供商的服务、相关费用。

(6) 其他供应：如软盘、电话、参考书、培训资料等。

具体资源要求取决于项目组和公司。因此测试计划工作要仔细估算测试软件的要求。

8) 测试人员分配

计划测试员任务分配是指明确测试人员负责软件的哪个部分、哪些可测试特性。如表 4-1 所示。

表 4-1　测试人员任务分配列表

测　试　员	测试员任务
AI	字符格式：字体、字体大小、颜色、样式
Sarah	布局：项目符号、段落、制表符、换行
Luis	配置和兼容性
Jolie	UI：易用性、外观、辅助特性
Valerie	文档：联机帮助、滚动帮助
Ron	压迫和重负

实际责任表更加详细，确保软件的每个功能都分配到人。每一个测试人员都清楚自己负责什么，而且有足够的信息开始设计测试用例。

9) 测试进度

关于测试计划的一个重要问题就是测试工作通常不能平均分布在整个产品开发周期中。有些测试以说明书和代码审查、工具开发等形式在早期进行。但在项目期间，测试任务的数量、人员和测试花费的时间不断增长，在产品发布之间会形成短期的高峰。如图 4-2 显示了典型的测试资源图标。

持续增长的结果是测试进度不断受到项目中先前事件的影响。如果项目中某一部分交给测试组时间晚了两周，而按照进度只有三周时间测试，结果会怎样？三周的测试在一周内完成？还是将项目推迟两周？这种问题成为进度危机。

使测试任务摆脱危机的一个方法就是测试进度避免僵化地规定启动和停止任务的日期。

而是根据测试阶段定义的进入和退出规则采用相对日期。那么显然测试任务依赖其他交付内容。单个任务需要多少时间也很明显。对比表 4-2 和表 4-3 两种不同方式。

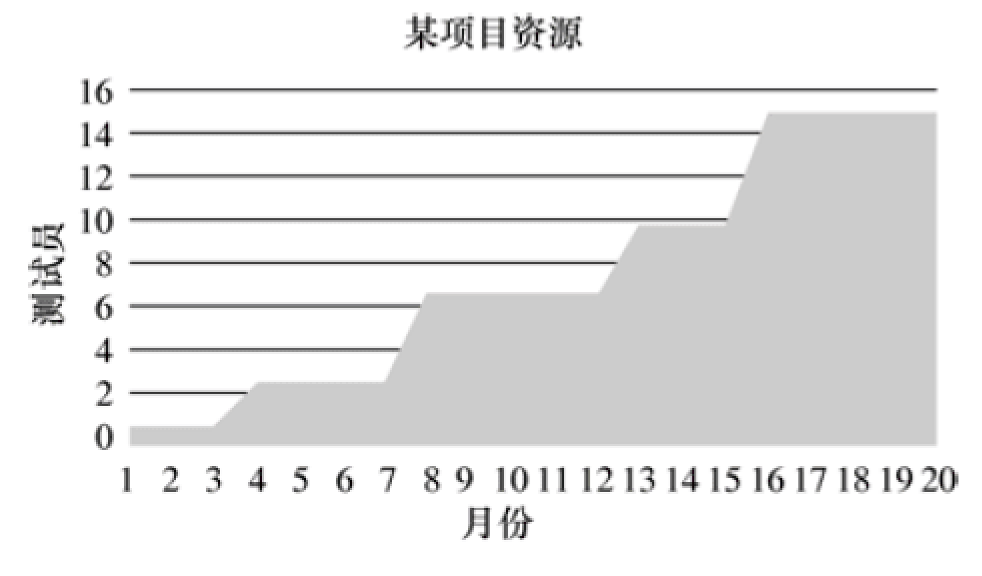


图 4-2　测试资源图标实例

很显然，这样的软件安排进度的方式会使得测试过程容易管理，项目管理员或者测试管理员最终负责进度安排，要求测试人员安排自己的具体任务。

表 4-2　测试任务时间计划表

测 试 任 务	日　　期
测试计划完成	3/5/2001
测试案例完成	6/1/2001
第 1 阶段测试通过	6/15/2001～8/1/2001
第 2 阶段测试通过	8/15/2001～10/1/2001
第 3 阶段测试通过	10/15/2001～11/15/2001

表 4-3　测试项目时间计划表

测 试 任 务	开 始 日 期	期　　限
测试计划任务	说明书完成之后 7 天	4 周
测试案例完成	测试计划完成	12 周
第 1 阶段测试通过	代码完成构成	6 周
第 2 阶段测试通过	Beta 构造	6 周
第 3 阶段测试通过	发布构造	4 周

10）测试用例

主要包括测试计划过程中决定采用什么方法编写测试用例，在哪里保存测试用例，如何使用和维护测试用例。

11）缺陷报告

通过哪些测试方法，发现了哪些 bug，这些 bug 又是怎样被解决的，解决之后有没有新的衍生 bug 等。

12）风险和问题

测试计划中常用而且非常实用的部分就是明确指出项目潜在的问题或者风险区域——

对测试工作产生影响之处。

假设有十几个测试新手，全部软件测试经验来源于书本，受命测试一个尖端科技的高危软件，这就是风险。也许没有人意识到某个新软件要测试全世界所有国家的语言，在项目进度中也没有留出足够的测试时间，这又是一个风险。

软件测试人员需要明确指出计划中存在的风险，并与测试管理人员和项目管理人员交流意见。这些风险应该在测试计划中明确指出，在进度中给予考虑。有些是真正的风险，而有些最终证实是可有可无的。要尽早指明，以免在项目晚期发现时，令大家感到惊慌失措。

3. 5W1H 法制定测试计划

下面介绍一种近些年被广泛使用的一种测试计划制定的方法。如图 4-3 所示。

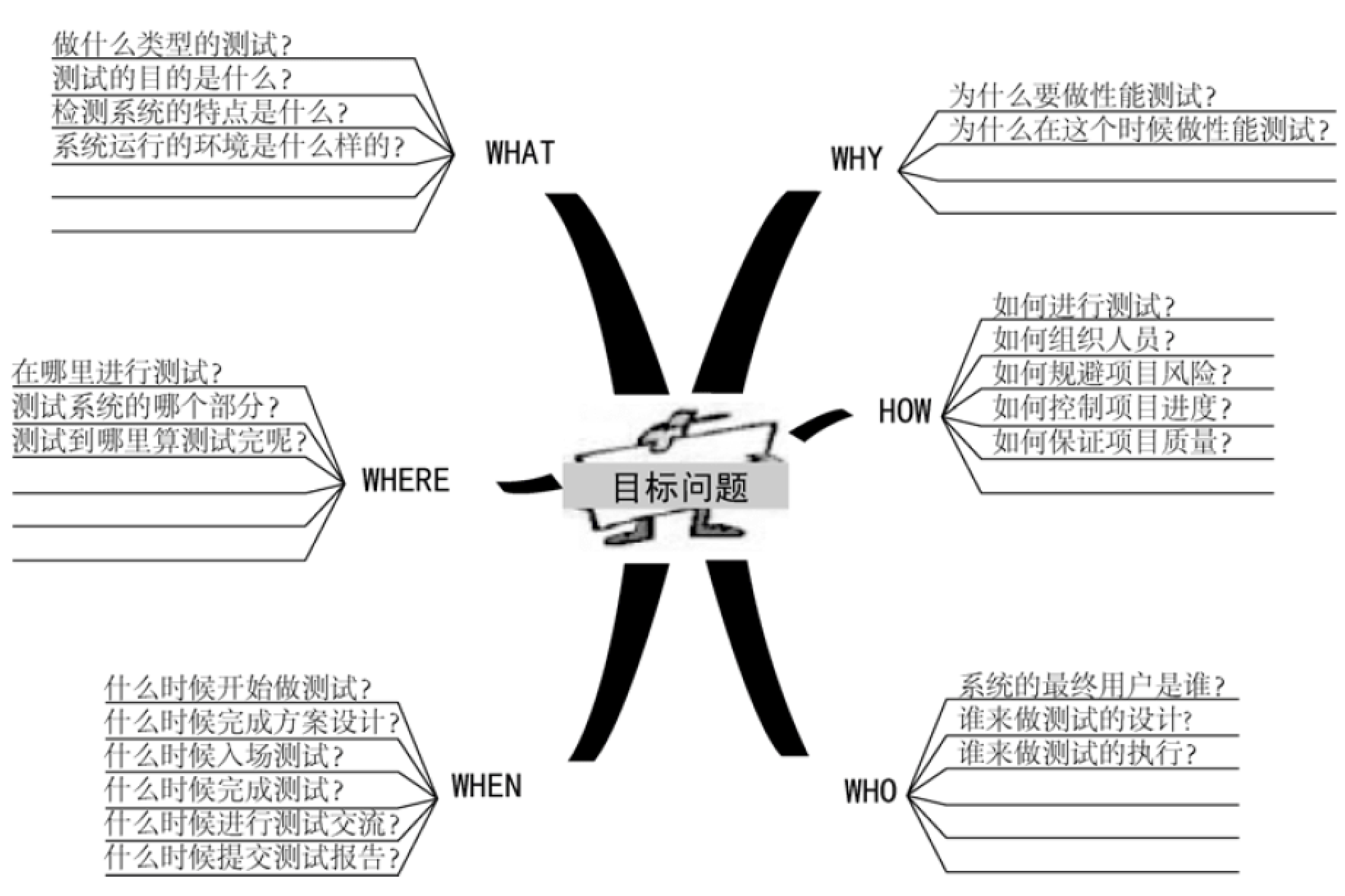


图 4-3　5W1H 测试分析方法图解

所谓 5W1H 就是 What、Where、When、Who、Why 和 How，见上图。

1) What

做什么类型的测试？测试的目的是什么？被测系统的特点是什么？系统运行的环境是什么样的？

2) Why

为什么要做这种类型测试？为什么要做性能测试？为什么在这个时候做性能测试？

3) Who

系统的最终用户是谁？谁来做测试的设计？谁来做测试的执行？

4) When

什么时候开始做测试？什么时候完成方案设计？什么时候完成测试？什么时候进行测试交流？什么时候提交测试报告？

5) Where

在哪里进行测试？测试系统的哪个部分？测试到哪里算测试完成？

6) How

如何进行测试？如何组织人员？如何规避项目风险？如何控制项目进度？如何保证项目质量？

4.3.3　测试规范

业界对测试规范的定义和应用层次有不同的理解。有的把测试规范认为是针对软件某具体功能的测试设计文档。而另一些则认为测试规范是为了一个特定的测试目的(例如，产品的验收等)，对被测软件产品或功能进行测试所需的有关文件。它规定性能特征要求、接口要求、测试内容、测试条件以及有关响应。本节介绍后一种测试规范。

软件测试是软件工程的重要组成部分，测试工作的质量直接影响软件产品的生命力。测试规范是测试团队的日常工作规范，主要侧重测试工作流程的控制，明确软件工程的各阶段测试团队应完成的工作、应如何完成以及完成的工作的标准。

软件测试规范一般包括团队职责、工作流程、缺陷管理、沟通管理、测试通过标准和引用文档等。这些内容在本丛书中均有详细介绍，本节不再赘述。

一份有效的、可行性高的测试规范至少包括下面的内容。

1. 软件测试规范的定义

软件测试规范就是对软件测试流程过程化，并对每个元素进行明确界定，形成完整的规范体系。软件测试规范是一个公司的测试标准，不仅是测试人员测试的准则，还是开发人员和测试人员达成的契约。一般来说，小公司或不正规的公司都不会书写这个，通常由测试经理来编写，测试工程师接触较少不太了解。

2. 软件测试规范描述的内容

一般来说软件测试规范描述的内容包括：测试目的、测试类别、测试过程、测试方法、测试用例、测试管理、测试文档、测试工具都要进行明确描述。

3. 具体内容包括

1) 测试计划规范

它包括测试计划模板的编写风格和测试计划的编写要求。如：测试进度估算、测试风险评估、测试人员安排和测试时间安排由什么来确定等内容。

2) 测试用例设计规范

它包含了测试用例的模板编写和测试用例的设计要求。如：测试用例设计人员、测试执

行时间、测试用例设计的优先级等。

3）测试工具使用规范

有了这个规范，测试人员就知道“项目进展”到什么程度，什么时候使用什么测试工具。建议把测试工具配置部分的“注意事项”也罗列在里中。比如使用 LoadRunner 做性能测试时，支持哪些常用的协议、使用那些脚本开发语言都要写清楚。

4）缺陷跟踪系统录入规范

主要是规范测试人员按照统一的要求递交缺陷到数据库。录入时，必须考虑缺陷录入的格式、录入的要素以及缺陷录入的“必填项”的要求等内容。

5）缺陷严重等级划分规范

有了缺陷严重等级划分规范，测试人员、开发人员和其他项目组成员对于测试缺陷就有了统一的标准，也不会因为某个缺陷由于严重等级的问题与项目组成员争论，从而提高了测试效率。

6）缺陷优先等级划分规范

优先等级规范的描述，有利于开发人员准确定位缺陷的优先等级标识，为开发人员修复软件缺陷和衡量产品质量提供参考。

7）缺陷分类规范

让测试人员准确对全部的缺陷，按“模块”进行准确分类，方便测试部门或质量部门对缺陷数量进行统计，并对软件质量进行评估，为软件是否允许发布提供重要的参考依据。

8）缺陷状态修改规范

要求测试管理系统的管理人员，根据不同的项目角色，准确分配缺陷管理系统的使用权限。如：开发人员不应该具备 Rejected、Closed、Suspended 的权限；测试人员不应该有 Fixed 的权限；还有如优先级、严重等级和版本等重要区域，都不允许修改。

9）缺陷递交流程规范

该规范是指测试人员“递交缺陷”、“缺陷公开”和开发人员修改缺陷后递交测试人员验证的流程，最好做成流程图的形式。

10）测试报告规范

它包括测试报告模板以及对测试报告编写的各种要求。如：测试报告包含的要素、测试缺陷分析的方法、分析手段以及缺陷分析应注意的问题，以上这些都要进行详细说明。

11）测试退出规范

软件测试到什么程度、满足什么条件，测试组织或测试项目就可以退出或停止。

12）软件测试类型规范

主要介绍测试的方法，包括单元测试、集成测试、系统测试、验收测试等测试方法。

13) 开发语言测试规范

比如需要测试的系统是使用 Java 开发的，你就要对 Java 的编程标准、初始化、面向对象编程、优化、javadoc 注释、线程、全局静态分析等语言基础有所了解，然后再有针对性地编写相关的测试规范。

14) 界面测试规范

一般来说目前流行的界面风格有三种方式：多窗体、单窗体以及资源管理器风格。我们在做界面测试时，同样要根据界面风格制定相关的测试规范，包括它的易用性、规范性、合理性、独特性、美观性和帮助设施等都要写入测试规范中。

15) 软件测试流程规范

软件测试的流程规范一般来说包括“测试项目确认流程”、“测试执行流程”、“测试策划流程”、“问题跟踪与测试关闭”等流程。

4.3.4 测试用例

本节主要讨论测试用例的格式，而非测试用例的设计。测试用例设计已在本套教科书中有专门的章节进行介绍。

软件测试用例的基本要素包括测试用例编号、测试标题、重要级别、测试输入、操作步骤、预期结果，下面将逐一介绍。

(1) 编号：测试用例的编号遵循一定的规则，比如系统测试用例的编号这样定义规则：PROJECT1-ST-001，命名规则是项目名称+测试阶段类型(系统测试阶段)+编号。定义测试用例编号，便于查找测试用例，便于测试用例的跟踪。

(2) 标题：对测试用例的描述，测试用例标题应该清楚表达测试用例的用途。比如“测试用户登录时，用户输入错误密码后软件的响应情况”。

(3) 重要级：定义测试用例的优先级别，可以笼统地分为“高”和“低”两个级别。一般来说，如果软件需求的优先级为“高”，那么针对该需求的测试用例优先级也为“高”，反之亦然。

(4) 测试输入：提供测试执行中的各种输入条件。根据需求中的输入条件，确定测试用例的输入。测试用例的输入对软件需求当中的输入有很大的依赖性，如果软件需求中没有很好地定义需求的输入，那么测试用例设计中会遇到很大的障碍。

(5) 操作步骤：提供测试执行过程的步骤。对于复杂的测试用例，测试用例的输入需要分为几个步骤完成，这部分内容在操作步骤中详细列出。

(6) 预期结果：提供测试执行的预期结果，预期结果应该根据软件需求中的输出得出。如果在实际测试过程中，得到的实际测试结果与预期结果不符，那么测试不通过；反之则测试通过。

软件测试用例的设计主要从上述 6 个领域考虑，结合相应的软件需求文档，在掌握一定测试用例设计方法的基础上，可以设计出比较全面合理的测试用例。

4.3.5　缺陷报告

在软件测试过程中，对于发现的每个软件错误(缺陷)，都要记录该错误的特征和复现步骤等信息，以便相关人士分析和处理软件错误。为便于管理测试发现的软件错误，通常要采用软件缺陷数据库，将发现的每一个错误输入到软件缺陷数据库中，软件缺陷数据库的每一条记录称为一个软件问题报告。

缺陷报告也算是一种测试文档。而且是最常用的一种文档。也是每个测试工程师必写的测试文档。本书第 5 章专门讨论软件缺陷和缺陷报告，因此本节只简单说明缺陷报告与其他测试文档间的异同以及缺陷报告的生命周期。

最早期的缺陷报告是用 Word、Notepad、Excel 等文档记录的，后来都采用在线的预先设置好的模板。

缺陷报告文档的几个特殊性如下：

(1) 只针对具体软件缺陷行为，也就是 Bug 具体信息；

(2) 有统一的在线模板；

(3) 缺陷报告的编写质量是衡量测试工程师技术水平的常用度量；

(4) 缺陷报告的信息直接关乎软件产品具体功能和设计行为；

(5) 缺陷报告是开发人员、测试人员、项目经理每天工作的主要对象；

(6) 缺陷报告的数量是所有软件测试项目衡量软件质量的重要指标之一。

缺陷报告的生命周期的如图 4-4 所示。从缺陷报告的生命周期可以看出，缺陷报告文档只侧重于缺陷本身，是测试工作最详细、最底层的对象。

编写缺陷报告可以参考以下 5C 准则。

(1) Correct(准确)：描述缺陷行为时要注意描述准确，指出症结所在；

(2) Clear(清晰)：描述缺陷的主题、步骤、错误之处和期望行为等的文字都要清晰明了；

(3) Concise(简洁)：尽量简化缺陷重现步骤，描述的文字要简单、易懂；

(4) Complete(完整)：描述意思完整；

(5) Consistent(一致)：用词要保持前后一致。

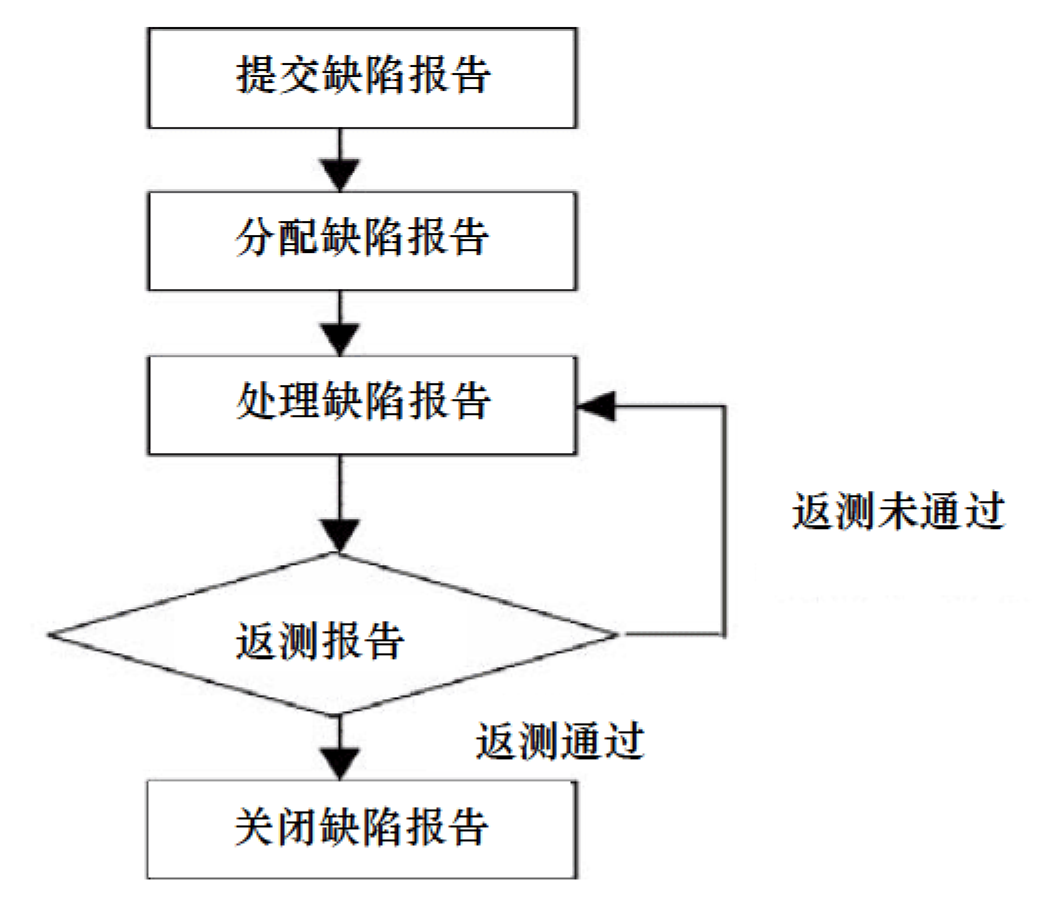


图 4-4　缺陷报告的生命周期

4.3.6　测试结果报告

软件测试结果报告是测试文档中重要的一种，是把测试的过程和结果写成文档，并对发现的问题和缺陷进行分析，为纠正软件存在的质量问题提供依据，同时为软件验收和交付打下基础。

编写一份优秀的软件测试报告并不是一件容易的事情，需要作者既对测试理论、测试技术、测试方法和测试等技术元素了如指掌，又要有出色的应用文写作功底。本节将介绍书写软件测试报告的一般方法。

首先确定报告的读者。项目团队中不同位置、不同角色的人对软件测试报告的关注侧重点有所不同。团队的管理层更关心项目经过测试后的整体状态，比如测试用例的通过率，不同优先级测试用例的通过率，尤其是高优先级的测试用例是否全部通过。当前软件产品的质量与基线的比较结果，是否达到预期的质量。当项目进行到一定阶段后，一些有经验的经理会根据历史信息对产品质量趋势进行预先判断。所以报告中应以简洁清晰的方式向管理层提供相关信息，比如图表、缺陷分布、缺陷趋势等。

团队成员更关心技术细节。一方面，他们可以根据测试报告，发现测试中是否还存在漏洞，可以将每个人的想法汇总起来，并给出补遗解决方案，帮助在下次测试中进行改进；另一方面，对于那些没有通过的测试用例，能够很容易地找到复现方法、出错信息等具体的信息。所以报告中要提供详细技术细节，保证团队中的一线工程师能够方便地获得相关信息。

在同一项目的不同阶段，软件测试报告的内容也有所不同。有些项目可能会要求每天或者每周都要提交一份测试报告汇报工作状态，报告今天或者本周完成的工作即可，比如测试的范围，发现了什么问题，提交了什么 Bug，项目进程等。而当一个迭代周期结束，执行全路径测试(Full Test Path)的时候则要非常正式，应包括明确标明通过或者失败的测试结果，以帮助管理团队判定是否可以结束本次迭代；此外，测试报告还应包括详细的测试结果和缺陷及其缺陷趋势分析等。如果明确得知当前质量状态满足不了质量标准，则应详细列出哪些质量标准没有被满足。比如，定义所有已知优先级为 1 的缺陷都已关闭才能结束此阶段，那么如果仍有缺陷状态为“活动”的话是不能结束的。报告中可以给出建议，以便管理层更好地实施决策。

书写测试报告的准则。首先，报告内容应是真实可靠的，测试工作应该严格按照测试计划执行，对结果的判定也应严格按照质量标准进行，不能因为受到某些方面的压力就放松质量要求，在报告中出现错误甚至虚假内容。当报告完成后，应反复进行审阅，保证没有错误。其次，充分考虑测试报告的读者，应使用准确、简洁的文风，保持测试报告有良好的格式。最后一个准则是行文保持客观、对事不对人，测试报告中描述出现的问题应关注问题本身，可以对缺陷的发生原因进行分析，但是不能进行过分的责任人追究，更不能进行人身攻击。比如，在分析缺陷分布的时候，一般可按功能模块、优先级、严重级进行分析，而不按照引入缺陷的工程师进行统计。

表 4-4 为读者提供一份软件测试报告模板做参考。为便于阅读，将模板放入一个表格呈现给读者。

表 4-4　测试报告模板实例

第 1 章　引言

1.1　编写目的

本测试报告的具体编写目的，指出预期的读者范围。

实例：本测试报告为 XXX 项目的测试报告，目的在于总结测试阶段的测试情况以及分析测试结果，描述系统是否符合需求(或达到 XXX 功能目标)并对测试质量进行分析。作为测试质量参考文档提供给用户、测试人员、开发人员、项目管理者、其他质量管理人员和需要阅读本报告的高层经理阅读。

Tips：通常，用户对测试结论部分感兴趣，开发人员希望从缺陷结果以及分析得到产品开发质量的信息，项目管理者更关心测试执行中的成本、资源和时间，而高层经理希望能够阅读到简单的图表并且能够与其他项目进行同向比较。

1.2　名词解释

列出本计划中使用的专用术语及其定义。

列出本计划中使用的全部缩略语全称及其定义。

缩写词/术语	英文解释	中文解释

1.3　参考及引用的资料

列出本计划各处参考的*经过核准*的全部文档和主要文献。

第 2 章　测试概述

2.1　测试对象

对测试项目进行简要的说明。

2.2　项目背景

对项目目的进行简要说明。必要时包括简史，这部分不需要脑力劳动，直接从需求或者招标文件中复制即可。

2.3　测试目的

对测试项目的测试目的进行简要说明，主要描述测试的要点、测试范围和测试目的。

2.4　测试时间

简要说明测试开始时间与发布时间。

2.5　测试人员

列出项目参与人员的职务、姓名、E-mail 和电话。

职务	姓名	E-Mail	电话
开发工程师			
CVS Builder			
开发经理			
测试负责人			
测试人员			

(续表)

2.6　系统结构

对系统的结构进行简要描述。参考系统白皮书，使用必要的框架图和网络拓扑图将会更加直观。

第 3 章　测试方法

测试方法和测试环境的概要介绍，包括测试的一些声明、测试范围、测试目的等，主要是测试情况简介。

3.1　测试环境

3.1.1　硬件环境

描述建立测试环境所需的设备、用途及软件部署计划。

“机型(配置)”：此处说明所需设备的机型要求以及内存、CPU、硬盘大小的最低要求；

“用途及特殊说明”：此设备的用途，如数据库服务器，Web 服务器，后台开发等；如有特殊约束，如开放外部端口，封闭某端口，进行性能测试等，也写在此列；

“软件及版本”：详细说明每台设备上部署的自开发和第三方软件的名称和版本号，以便系统管理员按照此计划分配测试资源；

“预计空间”：说明第三方软件和应用程序的预计空间；

“环境约束说明”：建立此环境时的特殊约束。如需要开发外部访问端口，需要进行性能测试等。

平台 1：S N					
机型(置)	IP 地址	操作系统	用途及 殊说明	软件及版本	预计空间
SUN450	10.1. .			oracle8.1.	2G
平台 2 IBM					
机型	IP 地址	操作系统	用途	第三方软件及	预计空间

3.1.2　软件环境

软 件 需 求	用　途

3.2　测试工具

此项目将列出测试使用的工具以及用途：

测 试 工 具	用　途
自动测试工具	

3.3　测试方法

简要介绍测试中采用的方法和测试技术。主要是黑盒测试，测试方法可以写上测试的重点和采用的测试模式，这样可以一目了然地看清是否遗漏了重要的测试点和关键块。

3.4　测试用例设计方法

简要介绍测试用例的设计方法，使得开发或测试经理等人员阅读的时候容易对测试用例的设计有个整体的印象，特别是一些异常的设计方法或关键测试技术，需要在这里进行说明。

(续表)

第 4 章　测试结果及缺陷分析

这是测试报告的核心，主要汇总测试各种数据并进行度量，度量包括对测试过程的度量和能力评估、对软件产品的质量度量和产品评估。

4.1　覆盖分析

4.1.1　需求覆盖分析

需求覆盖率是指经过测试的需求/功能和需求规格说明书中所有需求/功能的比值，通常情况下要达到 100%的目标。

需求/功能(或编号)	测试点描述	是否测试	重要等级	是否通过	备注

根据测试结果，按编号给出每一测试需求的通过与否结论。

需求覆盖率=测试通过需求点/需求总数×100%

4.1.2　测试覆盖分析

测试覆盖根据经过测试的测试用例和设计测试用例的比值，通过这个指标获得测试情况的数据。

需求/功能(或编号)	测试用例数	执行数	未执行数	通过数	失败数	备注

测试覆盖率=执行数/用例总数×100%

测试通过率=通过数/执行数×100%

4.2　缺陷统计与分析

对测试过程中产生的缺陷进行统计和分析。

4.2.1　缺陷统计

4.2.1.1　Bug 列表

这部分主要列出测试过程中的所有 Bug，并对其进行描述。

序号	BUG ID	描述	等级	模块	测试人员	开发人员

4.2.1.2　重要解决 bug 列表

这部分主要列出测试过程中产生关键的并且解决了的 Bug，对于重要的 Bug，需要对其产生的原因和解决方法进行分析说明。

序号	BUG ID	描述	等级	模块	测试人员	开发人员	Bug 分析

4.2.1.3　遗留 Bug 列表

这部分主要列出已经发现尚未被解决的 Bug，并对其进行描述，对于未解决的问题，需要在测试报告中详细分析产生的原因和避免的方法。

序号	BUG ID	描述	等级	模块	测试人员	开发人员	Bug 分析

(续表)

4.2.2　缺陷分析

本部分对上述缺陷和其他收集数据进行综合分析

4.2.2.1　缺陷综合分析

缺陷发现效率＝缺陷总数/执行测试用时

得出平均指标

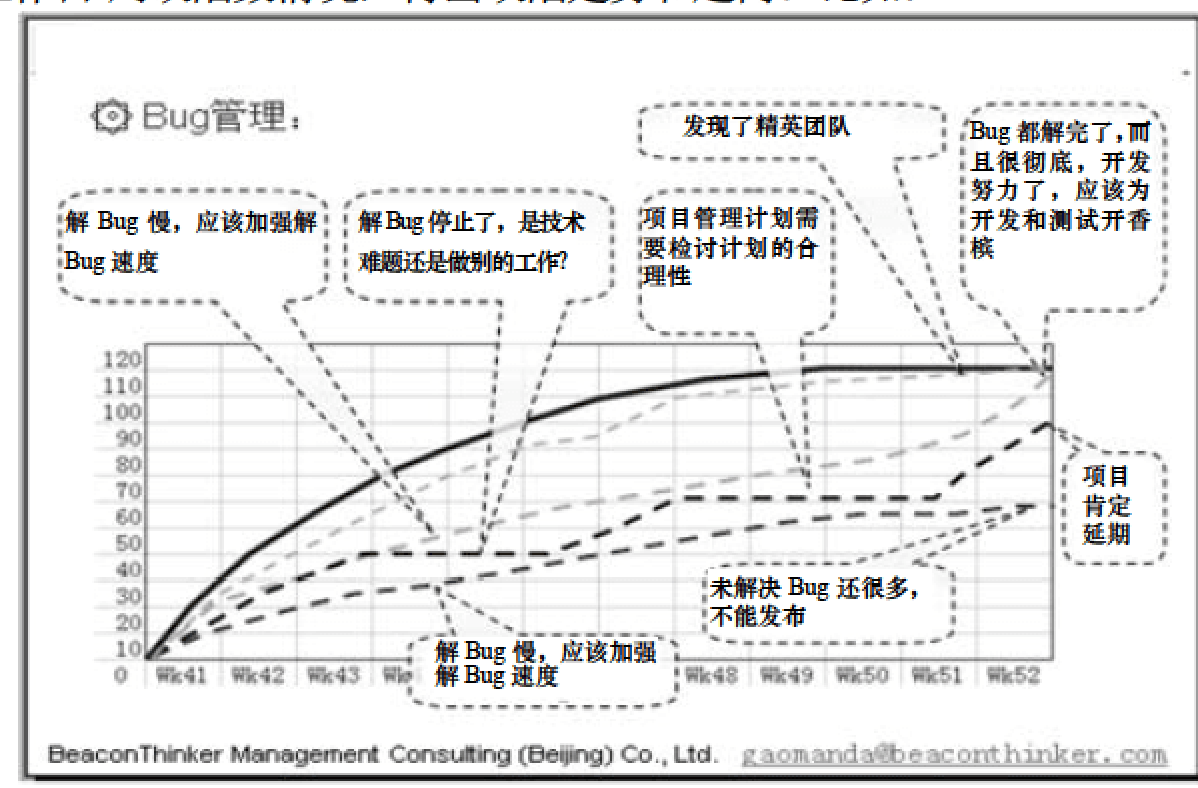
用例质量＝缺陷总数/测试用例总数×100%

缺陷密度＝缺陷总数/功能点总数

缺陷密度可以得出系统各功能或各需求的缺陷分布情况，开发人员可在此分析基础上得出那部分功能/需求缺陷最多，从而在今后开发中予以注意避免，并注意在实施时予与关注，测试经验表明，测试缺陷越多的部分，其隐藏的缺陷也越多。

4.2.2.2　测试曲线图

描绘被测系统每工作日/周缺陷数情况，得出缺陷走势和趋向。比如：



4.3　性能数据与分析

这部分简要地列出性能测试结果，并对测试结果进行分析说明，以说明是否符合软件需求。该部分也可以在性能测试报告中进行说明。

4.3.1　性能数据

记录测试输出结果，将测试结果数据表格、图表如实地反映到测试结果中。用于数据分析。例如：

短信数目(条)	时间(秒)	平均速度(条/秒)	最高速度(条/秒)	最低速度(条/秒)	IDLE 占用率平均，%)	MEM 使用率(平均，%)	CPU 使用率(平均，%)

4.3.2　测试结论

记录测试输出结果。用于数据分析。例如：

在分别处理 1 万条神州行和全球通的 MO 短信的情况下，短信处理速度为 400 条/秒。

测试结果对比：IAGW1.1 短信最大处理能力为 330 条/秒，本次 release 的 IAGW1.1 性能略有提高。各模块运行稳定。

(续表)

4.4　软件尺度

这部分主要是软件质量量度的一个尺度总表，主要是对上述分析的一个总结。

项　目	结　果	描　述
测试执行时间进度		
测试执行总天数		
测试准备时间		
测试总时间		
软件 Build 次数		
测试人力资源		
测试硬件资源		
测试项目总数		
自动测试项总数		
推迟测试项总数		
未测试项目总数		
测试用例总数		
自动测试用例总数		
成功测试用例总数		
发现错误总数		
修正错误总数		
已知错误总数		
测试执行时间细分 Accecpt Test Smoke Test Build First Regress Test First Build Second Regress Test Second Release Check		

第 5 章　测试总结和建议

这部分是测试报告中最关注的内容，主要是对测试过程产生的测试结果进行分析之后，得出测试的结论和建议。这部分是测试经理、项目经理和高层领导最关心的部分，因此需要准确、清晰、扼要地对测试结果进行总结。

5.1　软件质量

(续表)

说明该软件的开发是否达到了预期的目标，能否交付使用。 5.2　软件风险 说明测试后可能存在的风险，对系统存在问题的说明，描述测试所揭露的软件缺陷和不足，以及可能给软件实施和运行带来的影响。 5.3　测试结论 对测试计划执行情况以及测试结果进行总结，包括： 1. 测试计划执行是否充分(可增加对安全性、可靠性、可维护性和功能性描述)； 2. 对测试风险的控制措施和成效； 3. 测试目标是否完成； 4. 测试是否通过； 5. 是否可以进入下一阶段项目目标。 5.4　测试建议 对软件的各项缺陷所提出的改进建议，如：各项修改的方法、工作量和负责人、各项修改的紧迫程度、后续改进工作的建议、对产品修改和设计的建议、对过程管理方面的建议等。

4.4　测试文档管理

测试过程中的文档有：分阶段测试计划文档，测试流程文档，测试数据文档，测试参数设置文档和测试指南文档等。这些文档将会伴随着软件测试的各个阶段逐渐充实、完善，同时记载了整个测试的过程和成果。所有测试文档都有着重要保存价值和参考意义。因此测试相关的文档要妥善管理和维护。

高水平的测试文档管理有助于项目测试水平的提高。从内容上说，项目测试文档大致可以分为测试成果文档和测试过程文档两大类。测试成果文档作为项目可交付物的一个组成部分，其重要性自然不言而喻。测试过程文档主要记录了项目测试过程中的各种信息，为测试人员提供决策依据，以保证项目的顺利实现。另一方面，测试过程文档也是测试过程最宝贵的资产，通过对过程文档进行归纳和分析，可以对测试项目的成功经验和失败教训了然于胸，从而使后续测试运作更加有的放矢。

测试文档也驱动着测试过程。测试项目的阶段性成果是以测试文档形式体现的，因此，“测试项目的运作在一定程度上是由测试文档驱动的”这句话是有道理的。从测试文档的角度看，项目测试过程就是一个文档制作与执行的过程。在项目测试的过程中，每项工作的事前计划、事中测试记录、事后分析结果都要形成相应的测试文档，文档包括与项目相关的资源及其使用情况。

因此，测试文档是软件项目的一部分，没有正式的测试文档的活动，就不是规范的测试。测试文档的编制和管理在项目测试中占有突出的地位和相当大的工作量，高质量地编制、变更、修正、管理和维护文档，对于提高项目测试的质量和客户满意度有着重要的现实意义。

测试文档的管理可以注重三个方面：一是测试计划的评审，二是测试用例的评审，三是测试文档管理工具。下面逐一加以分析和说明。

4.4.1　测试计划的评审

好的测试计划是成功的一半，另一半是对测试计划的执行。前面已经描述了测试计划应该如何制定，但还要对测试计划进行管理和监控。一份测试计划投入再多的时间去做也不能保证能 100%完善，也不能保证 100%按照这份测试计划实施。对小项目而言，一份更易于操作的测试计划更为实用，对中型乃至大型项目而言，测试经理的测试管理能力就显得格外重要，要确保计划不折不扣地执行下去，测试经理的人际协调能力、项目测试的操作经验、公司的质量现状都能够对项目测试产生足够大的影响。此外，因为测试计划不是一成不变的，因此没必要把所有可能的因素都先包括进去，也没必要针对这种变化额外制定“计划的计划”，测试计划制定不能在项目开始后束之高阁，而是应该注意测试项目中的变化，实时进行思考和贯彻，根据现实修改，然后成功实施，这才能实现测试计划的最终目标——保证项目最终产品的质量。

测试计划编写完成后，一般要对测试计划的可行性、全面性以及正确性等进行评审。评审会由测试负责人主持，参加评审人员包括项目经理、软件开发团队、产品部门、市场部门等软件测试干系人。必要时甚至需要邀请法务等部门参加测试计划的评审。

测试计划的评审主要是根据一些规则依次检查，以下就列举两个不同风格的检查规则清单(如表 4-5 及表 4-6 所示)。在工作中可以参考这两个清单中的检查项，根据实际情况做适当的合并、增补或者删除。

表 4-5　测试计划检查项列表

编号	检查点/缺陷声明
01	是否根据测试中受影响的人的标识来定义测试计划的范围
02	测试计划中是否包含被测产品简介
03	是否在测试计划中定义了产品的重要质量指标，比如它的可靠性、运作正确性、稳定性、使用效率、使用便捷性等
04	是否在测试计划中定义了需要完成的可衡量的测试目标
05	测试计划中分配给测试团队的测试目标是否已划分优先级
06	测试计划中是否描述了关键的期望或假设，如所需的资源或测试预算等
07	在测试计划中，是否对测试工作中所涵盖的领域都有明确的定义。测试领域可以是数据、安装、用户界面、错误处理、性能、压力、实用程序、用户情况等
08	在测试计划中，是否对测试工作所涵盖的产品的功能有明确界定。描述测试的优先级，比如高、中、低
09	是否已经定义了分类来测试产品生命周期的不同阶段： ● 需求收集阶段 ● 设计阶段 ● 开发阶段 ● 稳定阶段

(续表)

编号	检查点/缺陷声明
10	是否为产品生命周期的每一阶段定义测试团队所需执行的活动
11	测试团队是否打算参加需求和设计的评估
12	测试团队是否要做可用性测试
13	测试小组将何时开始编写测试用例等
14	功能测试要进行多少轮测试
15	计划何时停止测试
16	是否定义了验收标准来验证产品质量过关并可以发布
17	是否已确定测试类型、完成标准、特别注意事项(如果有) 测试类型可以是： 功能测试 用户界面测试 负载测试 压力测试 配置测试 安装测试
18	是否已描述部署测试工具的类型
19	是否描述了测试计划，其中包括每个测试活动开始的计划日期
20	是否描述了整体项目计划，包含标识每个阶段的计划日期
21	是否定义了要成功执行该测试计划、需要的人数、人员的角色和职责、人员的技能细节等
22	是否定义了硬件需求，如测试组用来测试的工作站数量以及其配置
23	是否定义了软件需求，例如工作站的操作系统和其他普通软件，如微软的 Word 等
24	是否定义商业测试工具的需求及其需要的许可证数量
25	是否描述了产品安装的环境配置、描述测试所需的实例数
26	是否已定义沟通方式，例如测试团队如何将测试进度的报告提交管理者，将如何向上级报告问题和顾虑
27	是否描述了 Bug 报告和跟踪机制，即如何把 Bug 报告给开发人员，以及他们将如何跟踪 Bug 直至关闭
28	计划举行什么类型的会议，描述会议的目的，会议的参与者，会议时间安排和会议时长
29	要生成哪些类型的报告，描述它们的内容、目的和目标受众，生成报告的频度(周/次还是月/次)

表 4-6　质量计划检查项列表

编号	检查点/缺陷声明
01	质量计划是否控制项目实施
02	质量计划是否控制项目计划

(续表)

编号	检查点/缺陷声明
03	质量计划是否控制项目资源
04	质量计划是否控制项目审批
05	质量计划是否控制项目阶段，如项目的启动和结束
06	质量计划是否定义了所有必要的质量需求的类型
07	是否定义不同人员的责任
08	是否定义给不同的人员的委托授权
09	是否定义了正在开发的软件生命周期模型的类型
10	是否定义了在不同阶段采用的不同审查方法
11	是否定义了不同的测试方法
12	是否在质量计划中定义了适用于不同阶段的各种验证方法
13	是否在质量计划中定义了每个阶段的质量标准
14	是否明确定义了为特定配置管理行为的权限和责任
15	质量计划是否明确了权限和职责，用于监管验证你所购买的产品
16	质量计划是否明确了权限和职责，用于监管验证客户提供的产品
17	质量计划是否明确了权限和职责，用于监管验证内部开发的产品
18	质量计划是否明确了权限和职责，用于监管不合格产品权限和责任
19	当你的软件开发计划已经实施，你是否还会更新和完善质量计划
20	在质量计划实施之前，是否让所有组参与其审查和批准

4.4.2　测试用例的评审

测试用例的设计必须进行评审。测试用例应该由产品相关的软件测试人员和软件开发人员评审，然后根据评审意见更新测试用例。通过严格的测试用例评审，可以大大减少测试用例设计和实现中的错误，比如用例设计错误、用例设计冗余、用例设计不充分等。事实上，发现测试用例中的错误越晚，其造成的损失越大。如果在执行阶段发现，可能造成测试资源的浪费，如果在产品发布后发现有严重的错误或者遗漏，则可能造成有严重缺陷的产品被发布。

测试用例评审可以分为测试组内部和项目组评审。这两种评审，参与评审的人员范围不同，评审工作的内容和侧重点也不尽相同。

如果是测试组内部的评审，参与人员是测试组内的测试工程师、测试主管和测试经理，评审主要侧重于：

(1) 测试用例本身的描述是否清晰，是否存在二义性；

(2) 是否考虑到测试用例的执行效率。往往测试用例中步骤不断重复执行，验证点却不同，而且测试设计的冗余性造成了效率的低下；

(3) 是否针对需求跟踪矩阵且覆盖了所有软件需求；

(4) 是否完全遵守了软件需求的规定。这并不一定，因为即使再严格的评审，也会出现错误，应视具体情况而定。

如果是项目组内部的评审，参与评审的主要有项目经理、开发团队，必要时可能还会包括产品部门和市场部门。评审角度不同，评审的侧重点也不同：

(1) 收集客户需求的人员注重测试用例是否符合业务逻辑；

(2) 分析软件需求规格的人注重测试用例是否与软件需求规格要求一致；

(3) 开发负责人会注重你的用例中对程序的要求是否合理。

表 4-7 汇总了常见的测试用例评审的检查项，供读者参考。

表 4-7　测试用例检查项列表

序号	主要检查项
1	《需求规格说明书》是否评审并建立了基线
2	是否按照测试计划时间完成用例编写
3	需求新增和变更是否进行了相应调整
4	用例是否按照公司定义的模板进行编写
5	测试用例是否覆盖了《需求规格说明书》
6	用例编号是否和需求进行对应
7	非功能测试需求或不可测试需求是否在用例中列出并说明
8	用例设计是否包含了正面、反面的用例
9	每个测试用例是否清楚地填写了测试特性、步骤、预期结果
10	步骤/输入数据部分是否清晰，是否具备可操作性
11	测试用例是否包含测试数据、测试数据的生成办法或者输入的相关描述
12	测试用例是否包含边界值、等价类分析、因果图、错误推测等测试用例设计方法？是否针对需求不同部分设计使用不同设计方法
13	重点需求用例设计至少要有三种设计方法
14	每个测试用例是否都阐述预期结果和评估该结果的方法
15	需要进行打印的表格、导入、导出、接口是否存在打印位置、表格名称、指定数据库表名或文件位置；表格和数据格式是否有说明或附件
16	用例覆盖率是否达到相应质量指标

以上介绍了测试用例的评审方法，下面介绍在工程实践中如何进行测试用例评审。

1. 何时进行测试用例评审

一般会有两个时间点：第一，是在用例的初步设计完成之后进行评审；第二是在整个详细用例全部完成之后进行二次评审。如果项目时间比较紧张，尽可能保证对用例设计进行评审，提前发现其中的不足之处。

2. 参与评审人员

如前文介绍，测试用例评审分为测试团队内部评审和项目团队评审。具体参与人员和侧重点参考前文，此处不再赘述。

3. 评审内容

评审的内容有以下几个方面：

1) 用例设计的结构安排是否清晰、合理，是否利于高效地对需求进行覆盖。

2) 优先级安排是否合理。

3) 是否覆盖测试需求的所有功能点。

4) 用例是否具有很好可执行性。例如用例的前提条件、执行步骤、输入数据和期待结果是否清晰、正确；预期结果是否有明显的验证方法。

5) 是否删除冗余的用例。

6) 是否包含充分的负面测试用例。充分的定义，如果在这里使用 2&8 法则，那就是 4 倍于正面用例的数量，一个健壮的软件，其中 80%的代码都是在“保护”20%的功能实现。

7) 是否从用户层面来设计用户使用场景和使用流程的测试用例。

8) 是否简洁，是否便于重复使用。例如，可将重复度高的步骤或过程抽取出来定义为一些可复用的标准步骤。

4. 评审的方式

测试用例的评审方式可以比较严肃，也可以比较灵活。

最便捷的方式是两名或几名工程师坐在一起对测试用例进行评审。这种方式通常在测试团队内部使用，在一些敏捷开发团队中也是比较常用的测试用例评审方式。这种评审方式的好处是比较快捷，随时可以进行；不足之处是缺乏历史信息记录，只有较少的人员参与评审。比较适合影响范围较小的测试用例的评审。

另一种方式是评审发起人通过邮件等方式将测试用例评审“推送”给相关人员，相关人员再将评审意见回复给评审发起人。这样的评审方式是异步的，比较适合参与评审人员不能找到共同时间或者不在同一地点办公的情况。但是这种方式缺乏交互，当产生分歧时，解决分歧的效率低，而且评审发起人收集整理评审反馈的工作也比较困难。

最正式的方式是召开测试用例评审会议。评审发起人事先将需要被评审的测试用例文档分发给参与评审人员并约定评审会议的时间。在评审会召开前，评审发起人收集整理评审反馈并在评审会上进行充分讨论。这种方式最正式，效果最好。缺点是代价较大，无论评审发起人还是其他参与者，都需要付出大量时间和精力，而且还要磋商合适的时间。所以这样的评审会议，往往是在测试用例最终定稿前进行。

5. 评审结束标准

在评审活动中会收集到用例的反馈信息，在此基础上进行用例更新，直到通过评审。

4.4.3　测试文档管理工具

测试文档的管理主要包括文档的保存、分类、更新、发布和版本管理等。测试文档，一般来说数量较少，类型比较固定，不适合使用通用的文档管理工具进行管理。在当代软件研发过程中，越来越倾向使用专业软件研发管理系统对整个测试工作进行管理，测试文档的管理被集成到专业软件研发管理系统中，成为软件研发管理的一个有机组成部分。

惠普的 ALM 就是这样一款集成了测试文档管理功能的专业软件研发管理系统，本节就 ALM 的测试文档功能进行简单介绍。具体如何使用 ALM 将在本丛书中的实践部分中详细介绍，并配合相关的上机操作。

使用 ALM 进行测试管理包括四个步骤：

(1) 明确条件：分析你的应用程序并且确定测试条件。

(2) 测试计划：根据测试条件创建测试计划。

(3) 执行测试：在测试运行平台上创建测试集(Test Set)。

(4) 跟踪缺陷：报告应用程序中的缺陷并记录下整个缺陷的修复过程。

贯穿每一个阶段，通过产生详细的报告和图表去分析数据。下面分析每一个步骤。

1. 明确条件

1) 分析被测系统的功能和业务流程并且确定测试条件、约束等。

2) 确定测试范围：检查应用程序的文档确定测试范围和测试目标、策略。

3) 建立需求：构建“需求树”确定完全涵盖你的测试需求。

详细需求信息：为“需求树”中的每一个需求话题建立一个详细的目录，描述每个需求，给它分配一个优先级，如果需要的话还可以加上附件。

4) 分析详细需求：这些产生的报告和图表可以用于分析测试需求的覆盖率，检查需求以确保它们在测试范围内。

2. 编制测试计划

1) 定义测试策略：检查被测试系统、测试环境和测试资源以确认测试目标。

2) 定义测试对象：将被测试系统以模块或者是功能来划分，构造测试计划树细分为测试单元或对象。

3) 定义测试：每一个模块都需要确定其需要的测试类型，如功能测试、性能测试、安全测试等，在测试计划树中为每个测试点添加基本说明。

4) 创建需求覆盖：连接每个测试和测试需求确保覆盖率。

5) 添加测试步骤：可以通过在测试计划(树型结构)中添加一些步骤来进行手动测试，测试步骤描述了测试注意事项、检查点、每个测试的预期结果。

6) 自动测试：利用惠普 UFT 或者是第三方的测试工具实现自动化测试脚本。

7) 分析测试计划：根据生成的报告和图表分析测试计划数据，检查确定与测试目标是否一致。

3. 测试执行

1) 创建测试用例：在工程中定义不同的测试组来保证与不同的测试目标之间的一致性，可能包括：在一个应用程序中测试一个新的应用版本或一个特殊功能。确定每个测试集都包括哪些测试。

2) 运行时间表：为应用程序测试员分配测试任务和时间表。

3) 运行测试：手动或自动的执行测试用例。

4) 分析测试结果：观察测试结果旨在确定在测试运行中出现的错误是否已经被发现。生成的报告和图表可以帮助你分析这些结果。

4. 缺陷跟踪

1) 添加缺陷：报告在被测试系统中新发现的缺陷。在测试过程中的任何阶段，测试工程师、开发者、项目经理和最终用户都能添加缺陷。

2) 检查新的缺陷：检查新的缺陷和确定哪些缺陷应该被修复。

3) 修复活动的缺陷：修复那些经团队研究决定需要修复的缺陷。

4) 测试新的构建：测试在你的应用程序中新构建的部分，确认缺陷被修复。

5) 分析缺陷数据：在缺陷被修复前，生成的报告和图表的可行性分析，缺陷的趋势线(图)能帮助确定什么时候发布应用程序。

4.5　测试用例管理

随着软件技术不断发展，软件规模越来越大，复杂程度越来越高，测试领域也随之涌现出了各种理论、方法和工具，对测试管理工作也提出了更高要求。这其中很重要的一个分支便是测试用例管理工具，它主要解决的是测试过程中对测试用例的存储、分类、版本控制和团队协作等问题。

4.5.1　编写测试用例的挑战与应对之策

在传统软件研发流程中，测试用例一般保存在独立的文档中，或者填写到基于 Excel 等软件的电子表格中。这样的模式在中小型软件研发团队中可以起到较好的效果，较之心口相传甚至没有书面测试用例，完全凭测试工程师一时发挥进行测试的形式，在当时有其积极作用。但是到了当代，软件研发趋于专业化、大规模化和快速化，传统的独立(电子表格)文件形式的测试用例管理模式遇到了极大挑战。

1. 测试用例的存储安全

基于独立(电子表格)文件的测试用例管理方式，往往面临如何安全地存储测试用例文件的问题。如果存放在某位测试工程师的计算机或者某个共享文件夹中，团队中其他成员访问、更新这个文件都很麻烦，而且存在权限管理的难题，难以保证只有相应权限的人对这个文件进行操作。如果遇到硬盘故障等灾难性事故，测试用例管理文件难免丢失。

2. 测试用例难于分类与查询

基于独立(电子表格)文件的测试用例管理方式，在测试用例的分类和查询方面也存在不足。测试用例可以从很多维度进行分类，比如按优先级和严重级、按功能性测试和非功能性测试、按基于图像界面还是 API 等。如果使用电子表格，可以通过增加一列的方式来管理，但是这样做，随着电子表格的不断膨胀，无论维护还是查询测试用例文件都会增加难度。如果想依据一些复杂的条件对测试用例进行查询，也会异常麻烦。

3. 与测试需求的对应关系难以维护

基于独立(电子表格)文件的测试用例管理方式，通常测试需求也是采用类似的方法。一般来说测试需求和测试用例会分布在两张电子表格中，如本书第 2 章中所述，测试需求与测试用例之间不仅存在一一对应关系，还存在一对多、多对一甚至多对多关系。首先测试用例要覆盖所有测试需求，进而要求建立和维护测试需求和测试用例的对应关系。由于测试需求与测试用例之间不一定是一一对应，所以无论将测试需求和测试用例同时存储在同一张电子表格还是分成两份存储，建立和维护测试需求和测试用例的对应关系将是异常复杂的事。尤其当测试需求发生变化时，测试用例的更新和对应关系维护将成为一件非常令人头疼的工作。

4. 团队合作问题

基于独立(电子表格)文件的测试用例管理方式，在多人团队合作方面也存在一定障碍。主要表现在权限管理和同步操作方面。独立的电子表格文件，难以实现精确的用户及用户群组管理。比如最基本的权限管理至少应包括：用户或用户群组可阅读、可添加、可更新、可删除及对测试用例进行访问，独立电子表格文件虽然可以实现文件级的访问控制，但是无法对每个测试用例进行访问控制，更无法对某类测试用例进行访问控制。另一方面，当一位工程师对测试用例管理文件进行操作时，会影响其他工程师对该文件的操作，甚至无法打开这个文件。当团队规模小的时候，还可以互相协商解决，当团队成员数量变大，潜在的同时操作可能性加大，相互影响就无法完全避免了。

5. 测试用例的版本信息难以完整管理

基于独立(电子表格)文件的测试用例管理方式，在文档更新历史信息方面的弱点也很突出。最严重的是独立文件几乎无法保存用户编辑的历史信息，更无法做到测试用例一级的管理，即使借助版本控制系统也只能实现文件级的历史信息管理，依然无法做到测试用例级的历史信息记录与管理。这点作为测试用例管理系统来说是一个致命软肋。其他的问题也可能增加工程人员的工作难度，比如测试用例的 ID 管理，当有测试用例被删除，采用自增长式的 ID 管理模式可能会造成已存在的测试用例 ID 发生变化，而手工书写 ID 的工作量和出错率可能都比较高。

6. 难以实现测试用例的执行与结果管理

基于独立(电子表格)文件的测试用例管理方式，可以在测试用例管理文件中加入一列或若干列用以记录执行测试用例的结果。但如前文所述，当软件测试工作规模变大，参与执行

软件测试人员数量更多，同时对测试用例管理文件进行操作产生冲突的可能性大大增加，造成工作效率下降。

7. 测试用例与缺陷的对应关系难以维护

基于独立(电子表格)的测试文档管理，通常会把测试用例和缺陷分在两张电子表格中。当某个测试用例执行失败后，至少要新建一个缺陷用于跟踪；反之，当发现一个缺陷后，如果在已存在的测试用例中没有与其对应的测试用例，则需要新建至少一个测试用例作为回归测试的测试用例。无论哪种情况，都要跨越两张电子表格中维护测试用例与缺陷的对应关系，这种方式的工作量和操作的便捷程度都是难以接受的。

4.5.2　最佳测试用例特点

1. 最佳测试用例的设计原则

1) 依据原则：测试用例编写的主要依据是项目提供的需求说明书和相关技术规范文档；

2) 全覆盖原则：对于需求说明书和相关技术规范中要求的主要功能点进行全覆盖测试，要求所有功能均能正常实现；

3) 规范原则：所有测试用例的编写要求规范，对于所有被测的功能点，应用程序均应该按照需求说明书和相关技术规范中的给定形式，在规定的边界值范围内使用相应的工具、资源和数据执行其功能；

4) 全面原则：测试不仅针对系统功能特性进行测试，对系统的其他质量特性也进行全面的测试与评估。

2. 测试用例编写应该满足的具体量化要求

1) 对于用户经常使用、关系到系统核心功能、优先级别较高的功能点，测试用例应该达到 100%覆盖率；

2) 针对各个系统端到端的功能以及与其他系统的接口的测试应该达到 100%覆盖率；

3) 测试用例包括正常输入和正常业务流程测试，也包括对非法数据输入和异常处理的测试，且对系统非正常操作的测试用例应占到总数的 20%～30%；

4) 测试用例执行结果可以从覆盖率、执行率、通过率等几个方面进行分析和考察。测试用例覆盖率是指测试用例覆盖的功能与测试需求功能的比值；测试用例执行率是指已执行的测试用例数与测试用例总数的比值；测试用例通过率是指成功执行的测试用例数与测试用例总数的比值。

理论上，测试用例的覆盖率需要达到100%，也就是说，测试用例必须覆盖全部测试需求，否则测试用例的设计是不全面的，无法保证测试质量，需要补充或者重新设计相应测试用例。实际工作中只能在现有条件、时间、人力资源限制下尽可能达到最高的覆盖率。

测试用例执行率是衡量测试效率的因素，一般说来，在测试完成时测试用例的执行率也需要达到 100%，也可能因为某些特殊原因导致测试中断而没有全部执行测试用例，可针对具体的情况进行分析。测试用例通过率是衡量用例本身设计质量和被测软件质量的因素，对于未能

成功执行的测试用例，要分析是用例设计错误还是被测软件错误导致用例无法顺利执行。

3. 最佳测试用例的特点

1) 完整性

完整性是对测试用例最基本的要求，尤其是一些基本功能项上，如果有遗漏，那将是不可原谅的。完整性还体现在中断测试、临界测试、压力测试、性能测试等方面，这些测试用例也要能够涉及到。

2) 准确性

测试者按照测试用例的输入一步步测试完成后，要能够根据测试用例描述的输出得出正确结论，不能出现模糊不清的语言。

3) 简洁性

好的测试用例每一步都应该有相应的作用，有很强的针对性，不应该出现一些冗繁无用的操作步骤。测试用例不应该太简单，也不能够太过复杂，最大操作步骤最好控制在 10～15 步之间。

4) 清晰性

清晰包括描述清晰，步骤条理清晰，测试层次清晰(由简而繁，从基本功能测试到破坏性测试)。清晰简洁对测试用例编写者的逻辑思维和文字表达能力提出了较高的要求。

5) 可维护性

由于软件开发过程中需求变更等原因的影响，常常需要对测试用例进行修改、增加、删除等，以便测试用例符合相应测试要求。测试用例应具备这方面的功能。

6) 适当性

测试用例应该适合特定的测试环境以及符合整个团队的测试水平，如纯英语环境下的测试用例最好使用英文编写。

7) 可复用性

要求不同测试者在同样测试环境下使用同样测试用例都能得出相同结论。

8) 其他

如可追溯性、可移植性也是对编写测试用例的一个要求。

4.5.3　区分测试用例颗粒度

测试用例的粒度是指一个测试用例覆盖软件功能点的范围，覆盖面广被称为粒度粗(大)，覆盖面窄被称为粒度细(小)。

测试用例的粒度粗和细各有优缺点。测试用例粒度较细，好比网眼较密的渔网，测试用例捕捉功能失效的能力越强，测试的覆盖率(尤其是代码覆盖率)越高，设计精细的测试用例也有利于测试工程师理解和执行测试用例。但是测试用例粒度过细，也会导致设计、开发和执行测试用例消耗的时间和人力等资源过多，尤其是执行全路径测试的时间过长，可能导致软件版本发布的延期。另外，当软件产品的功能发生变化时，测试用例粒度过细，会导致审查和更新测试用例的工作量大大增加。测试用例粒度较粗，可以通过少数测试用例覆盖更多的功能点，测试工程师能够更专注于需要和业务逻辑从而避免繁琐的流程，提高测试执行的

效率，把握重点需求，避免陷入机械性的测试。但是测试粒度过粗，可能导致功能点虽然被覆盖但是测试并不充分(比如，等价类划分不完整、未考虑边界值等情况)、测试工程师执行测试困难、发现缺陷后不易定位缺陷起因等问题。在设计测试用例时应考虑以下因素。

1. 项目的进度

在项目时间紧张的情况下，往往留给测试人员的时间很有限，测试工作的重点是尽可能提高测试覆盖率，及早发现高严重级的缺陷，这种情况下测试用例的粒度可以放粗一些的，但是"粗"不代表随意，虽然可以放"粗"一些，但是要能明确测试要点和通过标准，分清优先级。而在项目时间宽裕的情况下，则尽量将测试用例写细一些，因为在写测试用例的同时，也能加深对产品的理解。此外，测试用例的粒度写得细，也能为后期的测试执行提供很大的帮助，为后期确定测试退出提供一定的数据支撑。

2. 软件工程师的情况

在项目实施过程中，可能测试用例是由具有较丰富测试经验的人来负责并主导编写的，而执行测试用例由初级测试工程师完成。对于初级测试工程师，测试用例往往对他们更快更好地理解产品并完成测试任务提供了非常大的帮助。因此，对于初级测试工程师，希望测试用例粒度细点是可以理解并认可的。对于资深测试工程师，如果他们既是测试用例的编写者，也是测试用例的执行者，出于种种原因，在设计和书写测试用例时粒度可能会偏粗些，而在执行测试用例时会进行一些探索性测试，尤其是经过一段时间磨合，较为成熟的测试团队可能会更倾向于将测试用例的粒度放粗一些。

3. 客户需求

在有些项目中，客户在验收产品的时候，会要求直接提出测试用例的设计要求以及其执行情况，这种情况下，无论项目时间安排是否紧张，都需要按照客户的要求来完成，因为这是项目质量需求的一部分。

4. 项目是否具有延续性

随着产品的多元化，产品的升级换代速度是很快的，换个 UI 或者组织架构，或者重新包装一下就是一个新产品。这种情况下，总会有些功能模块是一致的，测试用例也是可以复用的。因此，对于一些具有延续性的项目，测试用例的粒度可以尽量细一些，这样有利于知识的传承。而对于那些一次性的项目，测试用例粒度可以适当粗些，以不影响项目质量为准。

总之，测试用例的粒度并没有特定的标准，要依据项目实际情况而定。从技术角度看，测试用例粒度越细越好。在实际执行中，要依据项目的质量要求以及时间、人员等资源限制和实际情况综合考虑，适当妥协，以适用为原则，采用恰当的测试用例粒度，保证项目在既定预算内，按照满足预订质量标准的前提下按时发布。

4.5.4　测试用例管理建议

开发和维护测试用例是软件测试过程中的重要步骤之一，也是衡量软件测试质量的核心影响因素。本节从开发、执行和维护几方面对测试用例生命周期和管理过程进行分析，提出

了测试用例开发、维护的相关原则。

完整的测试用例生命周期过程，它通常有测试条件标识、测试用例设计、测试用例实现、测试用例的执行，以及测试用例管理等几个阶段组成。由于不同公司的质量方针和测试策略的不同，采用的测试用例过程可能会有所不同或者侧重点不同。图 4-5 是测试用例生命周期的瀑布结构。

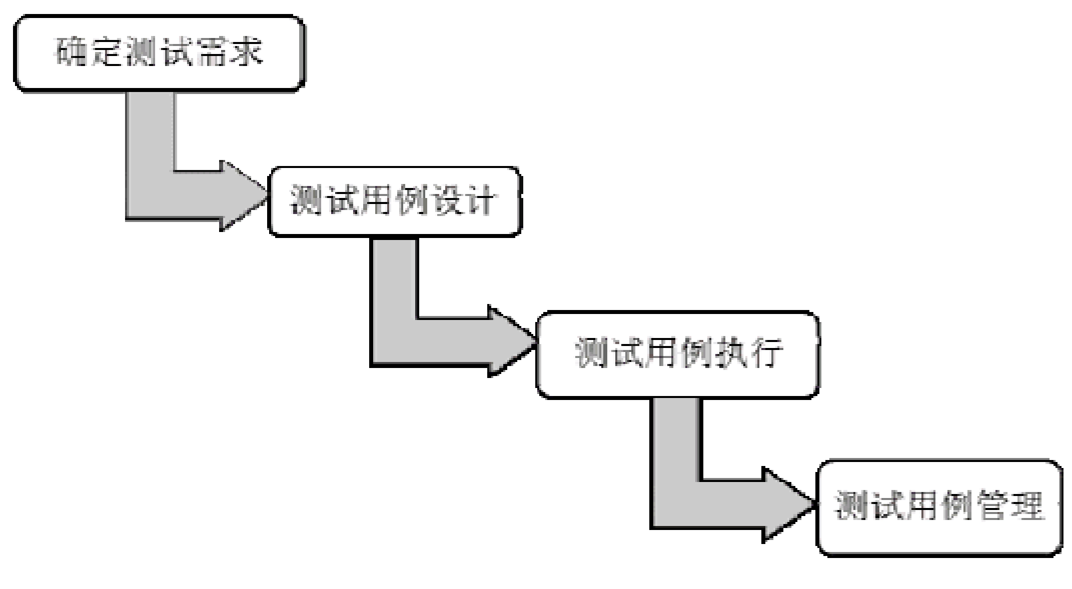


图 4-5　测试用例生命周期

测试用例通常是针对被测系统的功能模块来进行设计开发的。每个测试用例的设计都会涉及这四个过程。而且，测试用例的这四个阶段是有时间顺序要求的，例如：在设计测试用例之前，需要首先标识测试条件；在实现测试用例以前，需要设计测试用例。测试用例过程既可以是正式化的，比如对每个阶段的输出进行文档化，也可以是非正式化的。具体输出文档的详细程度，可以根据组织和项目的实际情况而定。

1. 确定测试需求

测试用例过程的第一步是确定测试什么(测试条件)，并且对测试条件进行优先级的划分。测试需求指的是可以通过测试进行验证的条目或者事件。针对测试系统，会有很多不同的测试条件。根据不同的测试条件，可以进行不同的测试类型分类，例如：功能测试、性能测试、可用性测试等。

在测试需求确认过程中，可以采用不同的测试技术，严格而系统地帮助测试人员获取测试条件，例如：黑盒测试中的等价类划分、边界值分析、因果图分析等，以及白盒测试中的语句覆盖、分支覆盖、条件覆盖等。

明确测试需求可以通过不同的方式来描述，比如通过简单的句子描述、通过表格的方式或者通过控制流图的方式等描述。

2. 测试用例设计

测试用例设计确定了如何测试已经识别的测试条件。测试用例指的是针对某个测试目标，而进行的一系列测试步骤。测试用例设计会产生一系列包含特定输入数据、预期结果和其他相关信息的测试用例。

测试用例设计的主要挑战是确定测试预期结果。为了确定测试预期结果，测试人员不仅需要关注测试输出，也需要注意测试数据和测试环境的后置条件。

假如测试依据可以清晰地定义，测试设计理论将是比较简单的。但是，测试依据通常情

况都是模糊不清的，至少描述是缺少覆盖率的。另外，即使清楚地描述了测试依据，测试输入和测试输出的复杂相互关系也会使得定义测试预期结果非常困难。假如测试用例没有测试预期结果，则测试用例对于测试结果的对错判断是毫无意义的，也无法提供有效的缺陷报告和增加客户对系统的使用信心。

测试预期结果可以是各种各样的，包括需要创建或者输出的结果，也可以是需要更新或者变更的结果，也可以是删除的结果。每个测试用例都应该清楚地描述测试的预期结果。也存在一些特殊情况，在测试用例中没有描述测试的预期结果，为了检查软件是否正确实现，必须对实际的输出结果进行仔细检查和验证。这需要测试人员具有被测系统相关的丰富的知识和经验，才可能对软件系统的测试输出作出正确评估。假如测试输出结果评估认为是正确的，就可以作为测试用例的期望结果。

同样，测试用例的设计，以及测试用例预期结果的设置应该和通过 V 模型左边的开发活动相对应。

3. 测试用例执行

测试用例执行过程包括准备测试脚本、测试输入、测试数据以及预期结果等。测试脚本指的是按照标准的语法组织数据或者指令，测试脚本一般保存在文件中，用于自动化测试。测试输入和测试期望输出也可作为测试脚本的一部分，也可保存在其他文件或者数据库中。

测试执行前，首先必须满足测试前置条件。比如一个测试用例需要用到文件中的一些数据，那么这个文件在测试执行时必须已经创建。测试前置条件也包括特定的测试硬件和软件，比如测试一个网络打印机，那么在测试之前，需要建立这样的测试环境，确保和打印机相关的网络是正常可用的。测试预期结果也可以保存在文件中，用于自动化测试。而对于手动测试，可以直接在测试用例中标识。

通过运行测试用例对被测系统进行测试。对手动测试来说，测试执行主要是测试人员参考测试用例的步骤来进行测试执行。测试人员输入测试输入、检查测试输出、比较测试预期结果和实际结果、记录在测试过程中发现的问题等。而对于自动化测试过程，测试执行可能是打开测试工具、运行测试用例脚本和测试脚本等，通过自动化方式来记录测试结果。

必须仔细检查每个测试用例执行的实际输出结果，根据测试预期结果来判断被测系统能否正确工作。有些测试结果的比较在测试执行中就可以进行，而有的测试结果需要在测试执行完成后才能进行比较。

假如测试实际输出结果和测试预期结果是一样的，那么认为测试用例执行是通过的。假如不一样，那么测试用例执行失败或者说软件系统存在问题。当然，这是非常简单的比较。更正确的说法是假如测试结果和测试期望不一致，需要进一步予以检查。比如可能是软件存在缺陷，也可能是测试用例本身存在问题，或者是测试用例中的测试预期结果存在问题，甚至是由于测试环境存在问题而引起的。所以，在比较测试结果时，需要从不同方面来确认具体的问题来源。

4. 测试用例管理

测试用例的管理包括测试用例组织、测试用例跟踪和测试用例维护。

1) 测试用例组织

任何一个项目，其测试用例的数目都将是非常庞大的。如何来组织、跟踪和维护测试用例是一件非常重要的事情。在整个测试过程中，可能涉及不同测试类型的测试用例。如何组织测试用例是测试成功与否的一个重要因素，也是提高测试效率的一个重要步骤。

可用不同方法对测试用例进行组织或分类：

(1) 按照软件功能模块组织：软件系统一般是根据软件的功能模块来进行工作任务分配的。因此，根据软件功能模块进行测试用例设计和执行等是很常用的一种方法。根据模块来组织测试用例，可以保证测试用例能够覆盖每个系统模块，达到较高的模块测试覆盖率。

(2) 按照测试用例类型组织：将不同测试用例进行分类和组织，也是一种常用的方法。比如可以根据配置测试用例、可用性测试用例、稳定性测试用例、容量测试用例、性能测试用例等对具体的测试用例进行分类和组织。

(3) 按照测试用例优先级组织：测试用例是有优先级的。对于任何软件，实现穷尽测试是不现实的。在有限的资源和时间内，首先应该处理优先级高的测试用例、用户最需要的功能模块或者风险最大的功能模块等。

在上面的三种测试用例组织方法中，按照功能模块进行划分是最常用的。不过，我们可以结合起来使用，比如在按照功能模块划分的基础上，再进行不同优先级的划分，甚至根据不同测试用例类型来进行划分和组织。

2) 测试用例跟踪

在测试执行之前，需要回答这样一些问题：哪些测试单元是需要测试的？有多少测试用例需要执行？如何来记录测试过程中测试用例的状态？如何通过测试用例的状态来确定测试的重点或者什么模块是需要进行重点测试的？

要回答这些问题，就需要对测试过程中的测试用例进行跟踪。测试过程中，测试用例的基本状态有三种：通过、未通过和未测试。根据在测试执行过程中测试用例的状态，实现测试用例的跟踪，从而达到测试的有效性。因此，测试用例的跟踪主要是针对测试执行过程中测试用例的状态来进行的，通过测试状态的跟踪和管理，实现测试过程和测试有效性的管理和评估。

(1) 测试用例执行的跟踪：在测试执行过程中，对测试用例的状态进行跟踪，可以有效地将测试过程量化。比如，执行一轮测试过程中，测试的测试用例数目是多少，每个测试人员每天能够执行的测试用例是多少，测试用例中通过、未通过、未测试的比例各是多少。这些数据可以提供一些信息来判断软件项目执行的质量和执行进度，并对测试进度状态提供明确的数据，有利于测试进度和测试重点的控制。详细的测试用例执行跟踪，可以参考测试用例评估章节。

(2) 测试用例覆盖率的跟踪：测试用例覆盖率包括了测试需求的覆盖率、测试平台的故障率、测试模块的覆盖率等。

测试用例的跟踪方式多种多样。具体需要跟踪组织的测试方针、测试过程和测试成熟度等。具体方法有：

(1) 没有任何记录：纯粹通过测试人员的记忆来跟踪测试用例。这种方法并不可取，除

非测试项目是基于个人开发的小型软件系统。

(2) 电子表格：使用电子表格对测试用例执行过程进行记录和跟踪是一种比较高效的方法。通过电子表格来记录测试用例执行状况，可以直观地看到测试的状态、分析和统计测试用例的状态以及测试用例和缺陷之间的关联状态，还有测试用例执行的历史记录等。这种测试用例的信息，可以为测试过程管理和测试过程分析提供有效的量化依据。

(3) 测试用例工具：最好通过测试用例的管理工具，对测试用例状态、缺陷关联、历史数据等进行管理和分析。工具不仅能够记录和跟踪测试用例的状态变化，也能生成测试用例相关的结果报表、分析图等，这样可以更高效地管理和跟踪整个测试过程。不过，工具的使用需要更高成本，并且需要专人进行维护。

3) 测试用例维护

测试用例并非是一成不变的，当一个阶段测试过程结束后，会发现一些测试用例编写的不合理，或者下个版本中部分模块的功能发生了变化，这都需要对当前的一些测试用例进行修改和更新，从而使测试用例具有可复用性。

一般在下面的情况下，可能需要修改或者更新测试用例有以下几类：

(1) 以前的测试用例设计不全面或者不够准确。随着测试的深入和对产品的熟悉，发现测试用例的步骤描述不够清楚或者描述的不够正确，甚至对系统需求的理解有误差。

(2) 测试过程中发现的一些问题，并不是通过执行当前的测试用例发现的。这时，需要增加测试用例来覆盖发现问题的一些步骤。

(3) 新版本中增加了功能或者功能的需求发生了变化。

(4) 随着版本的升级，有些测试用例需要删除。

软件产品的版本是随着软件的升级而不断变化的，而每一次版本的变化都会对测试用例集产生影响，所以测试用例集也需要不断变更和维护，使其与产品的变化保持一致。以下原因可能导致测试用例变更：

(1) 软件需求变更：软件需求变更可能导致软件功能的增加、删除、修改等变化，应遵循需求变更控制管理方法，同样变更的测试用例也需要执行变更管理流程。

(2) 测试需求的遗漏和误解：由于测试需求分析不到位，可能导致测试需求遗漏或者误解，相应的测试用例也要进行变更。特别是对于软件隐性需求，在测试需求分析阶段容易遗漏，而在测试执行过程中被发现，这时需要补充测试用例。

(3) 测试用例遗漏：在测试过程中，发现测试用例未覆盖全部需求，需要补充相应的测试用例。

(4) 软件发布后，用户反馈的缺陷：表明测试不全面，存在尚未发现的缺陷，需要补充或者修改测试用例。

对于提供软件服务的产品，其多个版本常常共存，而对应的测试用例也是共存的，而且测试用例需要专人定期维护，并遵循以下原则：

(1) 及时删除过时的测试用例。需求变更可能导致原有部分测试用例不再适合新的需求要求。例如：删除了某个功能，那么针对该功能的测试用例也不再需要。所以随着需求的每一次变更，都要删除那些不再使用的测试用例。

(2) 及时删除冗余的测试用例。在设计测试用例时，可能存在两个或者多个用例测试相同内容，降低回归测试效率，所以要定期整理测试用例集，及时删除冗余的测试用例。

(3) 增加新的测试用例。由于需求变更、用例遗漏或者版本发布后发现缺陷等原因，原有的测试用例集没有完全覆盖软件需求，需要增加新的测试用例。

(4) 改进测试用例。随着开发工作进行，测试用例不断增加，可能会出现一些对输入或者运行状态比较敏感的测试用例。这些用例难以重用，影响回归测试的效率，需要进行改进，使其可重用可控制。总之，测试用例的维护是一个长期过程，也是一个不断改进和完善的过程。

4.5.5　测试用例管理工具

在当代软件研发过程中，通常使用基于数据库的软件研发管理系统，测试用例管理是其中的一个主要功能模块，它与其他模块一起协作帮助软件研发团队更加高效地完成软件研发工作(如图 4-6)。

测试用例管理工具一般应包括如下功能。

1. 测试用例 ID 管理

管理工具可以自动为新建的测试用例分配一个唯一的 ID 作为标识符。一般情况下每个测试用例的 ID 不会改变，当测试用例被删除后，该用例的 ID 也不会被重新分配给其他测试用例。

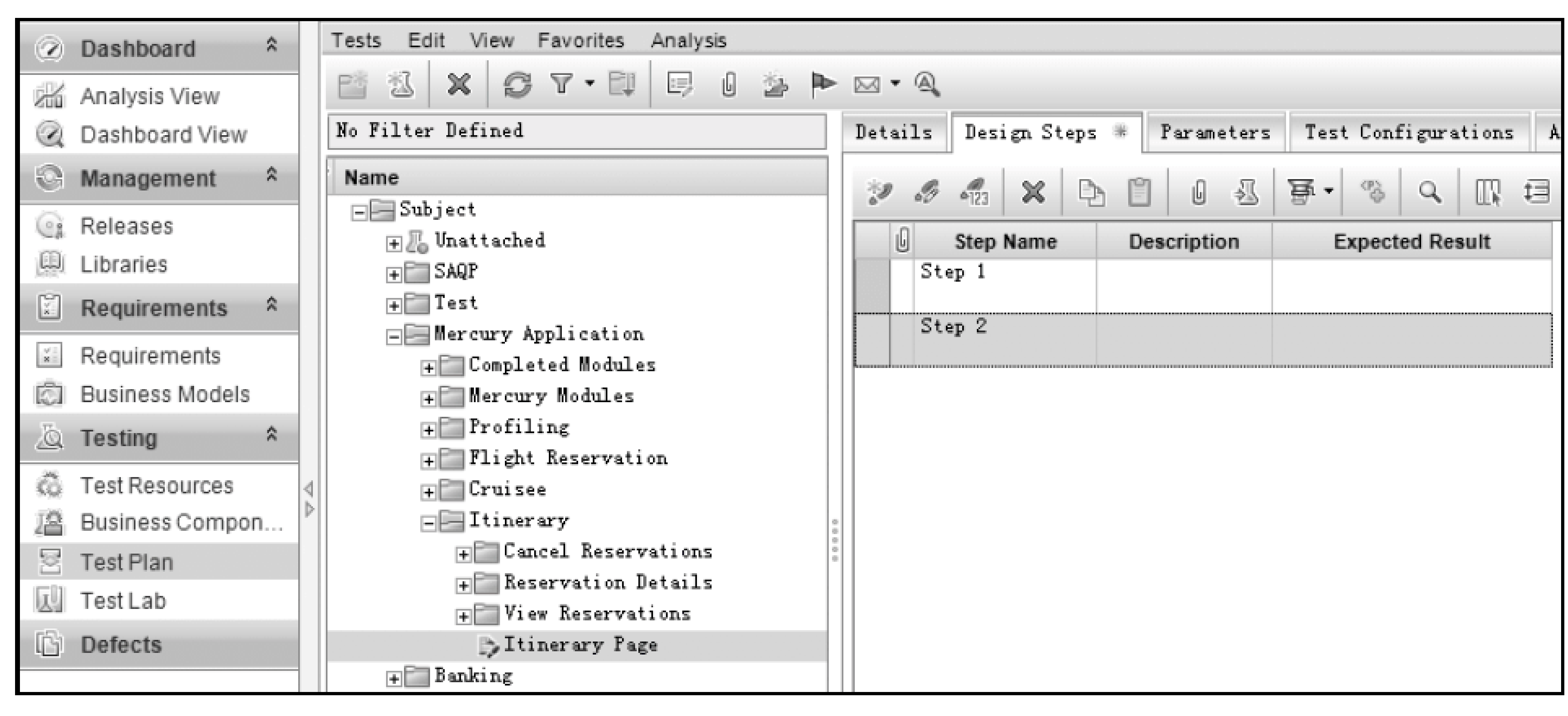


图 4-6　惠普 ALM 测试用例管理工具实例图

2. 测试用例的维护

包括提供测试用例模板和新建、删除、修改等。

3. 测试用例分类管理

测试用例需要通过不同维度的分类进行管理。比如按照功能点或测试需求分类。按照测试类型，分为功能性测试用例、性能测试用例、压力测试用例、安全测试用例、国际化本地

化测试用例等。按执行测试用例的方式，可以分为自动化测试用例和手动测试用例。还可以按优先级分类。这些分类标准可以灵活制定，甚至可由用户自行制定。测试用例的分类管理通常是在数据库每个测试用例的条目中加入标签字段来实现。

4. 用例的导入导出

目前很多公司还在使用 Excel 来书写和保存测试用例，如果一家公司准备采用一套测试管理系统，将这些用例手工导入将是一项繁重工作。因此测试管理工具需要能够将 Excel 里面的用例导入到系统。同样也能将测试用例导出为 Excel 格式的文件。从数据库导出 Excel 的功能还是比较好实现的，在 Excel 的导入功能方面，可通过 Excel 的 VBA 编程自动实现数据的获取，并且可以更新回系统中，这样会更加便捷。

5. 用例搜索功能

基于数据库的测试用例管理工具可以提供方便快捷、功能强大的测试用例搜索功能。对于测试用例较多的大型软件测试项目而言，这可以极大地提高测试工程师的工作效率。

6. 提供测试需求、测试结果和缺陷的对应关系

当代软件研发管理系统一般都将提供测试需求、测试结果、缺陷和测试用例的管理集中在一个系统中，并通过数据库进行管理，与独立的电子表格文件方式相比，就较容易实现这些要素之间的对应关系，便于测试工程师以及团队其他成员对这些要素进行管理。

惠普的 ALM 就是这样一个现代化的软件研发项目管理系统。

4.6　测试文档最佳实践

如前所述，测试文档对于项目管理的作用是不容置疑的，但测试文档的管理却又通常是项目管理中最容易忽略的。在测试文档管理中应该要注意以下几个方面：

1) 建立测试文档管理制度。重点应体现为两点：①要对测试文档的名称、标识、类型、责任人、内容等基本内容做出事先安排，给出测试文档总览表。②制定各种测试文档的管理程序，如批准、发布、修订、标识、贮存、传递、查阅等，为测试文档配置管理铺设一个良好的基础平台。

2) 文档版本管理非常重要。版本混乱是测试文档的一个致命伤，测试文档的有效管理必须实行版本控制。

3) 创建测试文档库的访问规则，这是文档管理的重要环节。访问规则确定谁可以访问、阅读、升级文档以及谁可以在文档库中添加文档。同时，文档库还应定期进行检查，检查哪些文件需要进行存档或者哪些旧文件需要进行清理，以确保文档管理符合项目测试组的需求。

4) 使用工具管理文档。对于一个大型的项目测试，整个测试周期中都会有大量文档。测试文档内容也在不断变化中，有连续的、承前启后的、新增加的、也有的是废除的。这可能需要一个统一的文档管理工具，分门别类统一存放管理各种测试文档。

5) 编写缺陷报告的建议：①要多读优秀的缺陷报告，学习最佳实践；②每个缺陷报告尽量截取图片和 log 来帮助开发人员快速定位问题；③对重现步骤自己要多执行几遍，确保开发人员可以再现缺陷；④缺陷报告要客观得体，不要包含自己的主观情绪。

习题与思考题

1. 软件测试文档的重要性和必要性？
2. 编写缺陷报告的 5“C”原则是什么？
3. 常用的测试文档包括哪些？
4. 软件测试用例的基本要素包括哪些？
5. 最佳测试用例的设计原则？
6. 测试用例的颗粒度指的是什么？

第5章　软件缺陷管理

软件测试是评估软件产品质量的最关键手段之一，而且软件测试的重要目的是尽早和尽多地发现软件中存在的缺陷，并尽快将缺陷修复。尽早发现和修复缺陷是软件测试的基本原则之一，也是提高软件产品质量和降低成本的重要手段之一。因此，对软件测试过程中的重要输出，即软件缺陷的管理就变得非常重要了。本章结合实例，在介绍缺陷管理基本概念基础上，重点从“管理”的角度介绍缺陷管理的意义、角色分工、缺陷跟踪管理体系和缺陷报告中要考虑的因素，并对缺陷的统计与分析进行了初步介绍。本书的理论部分介绍缺陷管理应注意的 7 个重点，而本书的实践内容部分则结合惠普公司的测试管理工具 ALM 详细介绍实际工作中怎样实施缺陷管理。

作为应用型软件人才，学习完本章后，学员应具备以下能力：理解软件缺陷管理原则，根据软件缺陷管理的一般思想，快速掌握应用相关工具的使用，书写清晰明了的缺陷报告，可针对缺陷进行初步的统计与分析。

5.1　缺陷管理意义

初接触软件测试的读者可能认为缺陷的管理是项目经理或者测试管理人员的职责，与自己无关。其实测试人员恰恰是参与了缺陷的发现、报告、会审(Triage)、解决、验证和关闭的全过程。深入理解缺陷管理的流程是做好本职工作的重要基本技能之一，也是在职业生涯进一步发展的基础。如果做了三年甚至五年的测试人员，连缺陷生命周期都不能完整的掌握，是很难承担更富有挑战的职位的。

做好缺陷管理的目的在于以下十方面：

(1) 通过推广专业的技术找到隐藏的缺陷；

(2) 通过准确的文档报告缺陷；

(3) 通过良好的沟通使缺陷尽快解决；

(4) 通过良好的措施减少和预防缺陷的发生；

(5) 通过合适的工具管理缺陷报告；

(6) 通过优化的流程推进缺陷的生命周期；

(7) 通过缺陷数据的分析及时找到问题和根源；

(8) 通过角色的分工促进团队协作；

(9) 通过明确的分级确保优先的处理；

(10) 通过缺陷三方会审，一起做出正确决策。

在不同的公司、组织(甚至同一公司的不同项目间)，缺陷管理的具体流程、缺陷状态定

义和管理风格都可能不尽相同，但有很多相似之处。理解一套完整的缺陷管理要素有助于理解其他项目和公司的体系。

5.2　缺陷跟踪管理体系

传统上，缺陷管理更多靠人工跟踪和简单的工具。比如白板、邮件或独立的 Excel 表格。随着时代的发展，软件研发的规模越来越大，参与人员越来越多，甚至出现了同一项目的几个团队分属不同地域不同时区的情况。传统的人工跟踪和简单的工具已经跟不上软件研发过程标准化的进程了，于是出现了专业的缺陷跟踪工具，这些工具大部分都基于数据库系统，后文将进行详细介绍。

5.2.1　缺陷跟踪工具

在测试过程中发现的缺陷需要从包含缺陷发现、缺陷提交、缺陷分类、缺陷修复、解决方案验证的整个过程进行跟踪。为了保证有效且及时地解决测试过程中发现的缺陷，组织内部需要建立一套完整的过程和规则对缺陷进行跟踪和管理。详细记录缺陷相关信息的缺陷报告的结构可以参考 IEEE Std 829-2008。缺陷管理系统是用于跟踪和管理缺陷的专业软件系统，软件工程团队使用这个系统记录关于缺陷的各种专业信息，如标题、详细描述、严重级、优先级、发现者、负责人、状态等。绝大部分现行的缺陷管理系统都基于数据库的 C/S 或者 B/S 架构，可以多人同时对缺陷进行浏览和修改等操作。

在基于数据库的缺陷管理系统大规模应用前，白板、邮件、单独的类似于 Excel 的表格文件都曾用于缺陷管理，但它们各有致命的弱点。而使用基于数据库的缺陷管理系统对于改进软件开发流程、提高软件开发团队的效率有极大帮助。大部分缺陷管理系统都具备至少几个特点：

1. 简单易用

大部分缺陷管理系统都有友好的界面，提供专业缺陷模板、强大的组合条件查询功能和统计分析功能，测试工程师可方便地创建、修改和查询缺陷，项目管理人员可以方便地根据历史和当前缺陷的状态、趋势和分布等数据，对项目状态进行评估并制定合理的工作策略。

2. 可设置性强

大部分缺陷管理系统都可以根据项目的需要，对所需组件、缺陷模板、缺陷管理流程等进行定制。

3. 可靠性高

不同团队对缺陷跟踪常常有不同的需求和要求。能自由添加字段来存储更多信息，能选择哪些字段是系统必需或可选的都很重要。举例来说：当存入缺陷时，有些团队只要求输入标题、描述、严重性和版本修改号。其他团队也许要求额外的信息。

4. 利于团队合作

相比于单独的 Excel 文件，专业的缺陷管理系统更利于团队合作，可以多人同时对缺陷跟踪系统进行操作，保证所有工程师都及时分享同样的信息。单独的 Excel 文件也容易产生不同版本的问题。

5. 可以自定义强大的缺陷统计报告

专业的缺陷管理系统不但提供专业、强大的统计功能，而且提供定制搜索条件的功能。使用者(尤其是项目管理者)可以方便地掌握项目的进展情况并对未来进行预测。

5.2.2　HP 测试管理软件

HP 测试管理软件系列中的软件 ALM 包含缺陷管理功能，其产品的前身 Test Director 至今还有很多公司使用。ALM 相比 Test Director 的改进是把 Test Director 转移到了 J2EE 平台上，除了内建的 JBOSS 以外，还支持 Weblogic、Jboss、Apache 等主流 Webserver，具有良好的跨平台的特性。对 UFT/WinRunner 之类的 HP 系列产品的支持。惠普测试管理工具 ALM 的缺陷管理功能已经有缺陷报告数据库设计和很多简单易用的模板。见图 5-1。在本书后面的实践内容部分将详细介绍 ALM 怎样实施缺陷管理。

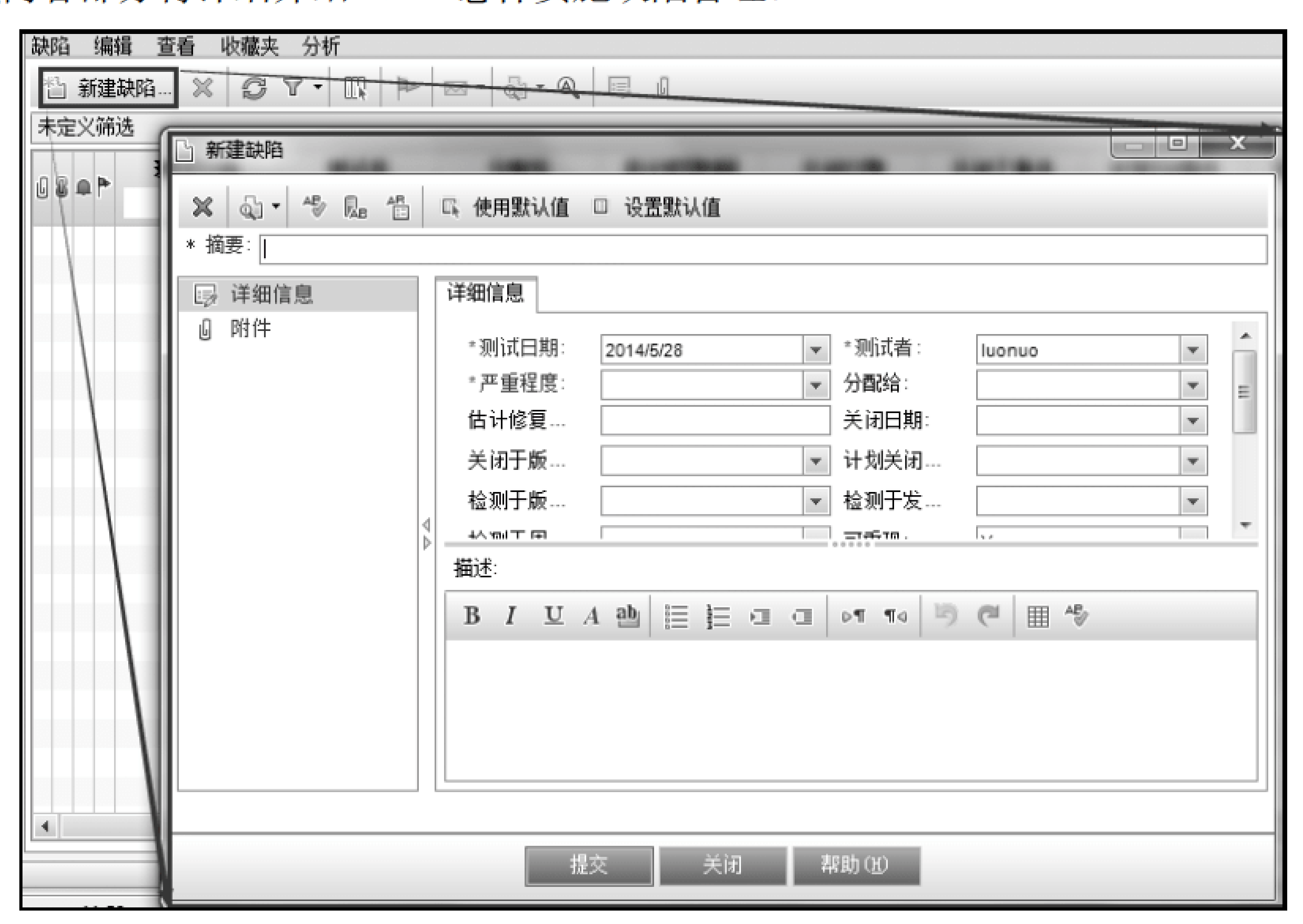


图 5-1　新缺陷报告界面

5.2.3　缺陷的关联与依赖关系

绝大多数缺陷管理工具设计都应有关于依赖关联(Dependency)的功能模块。对于依赖关联要注意以下几方面：

1. 依赖关联

要考虑一个 Bug 的解决常依赖于其他 Bug。例如：程序员 B 的缺陷只有等到程序员 A 的缺陷解决后才能解决；一个版本中的某个缺陷依赖前一个版本中的缺陷，甚至一个产品的缺陷依赖于其他产品的缺陷。依赖关联的处理确实比较复杂，不过一般使用的比例比较小。如果缺陷间存在依赖关系，那么处于被依赖(或者父依赖)的缺陷应尽快加以解决。

2. 重复关联

如果其中一个缺陷实际上是另一个缺陷的重复缺陷，就需要在这两个缺陷间建立起重复关联。我们要求测试人员尽量不要提交重复的缺陷，如果一旦提交了重复的缺陷，则在这组重复的缺陷间建立“重复关联”，一旦主缺陷解决，则所有的其他重复缺陷(从缺陷)也就都得到了真正的解决。

3. 相关关联

相对于依赖关联和重复关联，相关关联的概念要简单很多。同一个数据库中，当一个缺陷与其它缺陷存在一定的关系，但却并非依赖关系或重复关系时，就可以为它们建立相关关联。相关关联的各个缺陷之间不存在任何依赖关系，这意味着可以独立地处理各个缺陷。

4. 重复关联和附件

前者是因为各个测试很有可能登记了重复的缺陷，后者是因为有时文字描写不清楚，上传一张图片效果可以好很多。当然给出文件的链接也是好的。如图 5-2 所示，惠普缺陷管理工具中支持关联的使用。

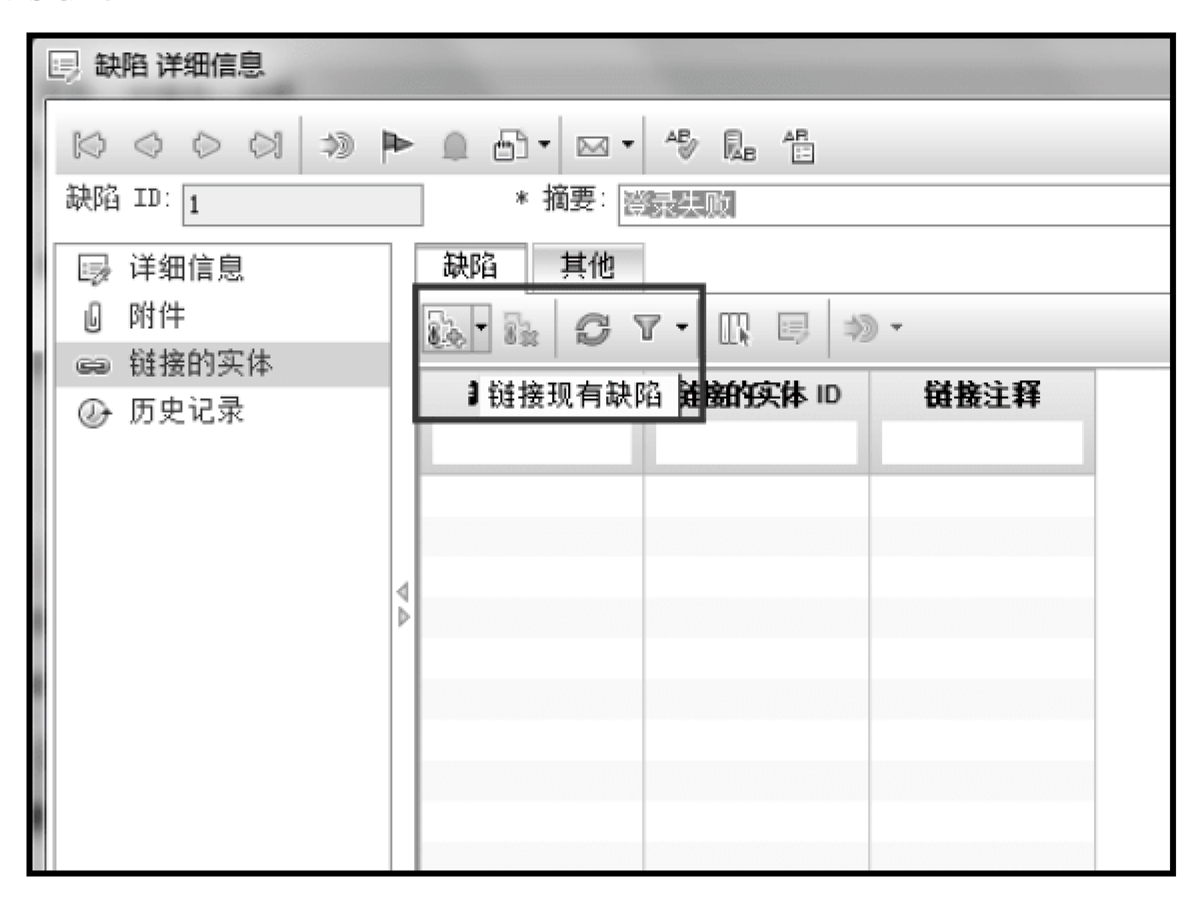


图 5-2　惠普缺陷管理工具中关联的使用

有些功能更强大的缺陷管理工具甚至将缺陷和相应的测试用例关联起来，将缺陷与源代码管理中的 Change Number (或者文件号)关联起来。当然，一个缺陷管理工具最基本的操作就是 New、Resolve、Close、Assign 以及 Query。能把这些最基本的操作做好，软件产品的质量已经会有很大的提升了。

5.3　缺陷管理责任分工

对现有项目团队进行评估和诊断、对其项目的软件缺陷进行有效治理以及规避战略决策失误是企业缺陷管理的关键问题。人力、财务和质量是企业内部控制的主要方面。如图 5-3 所示，开发和测试都有各自的生命周期，因此在一个团队协作中分清职责并落实到人，落实到具体任务是缺陷管理的重要一环。

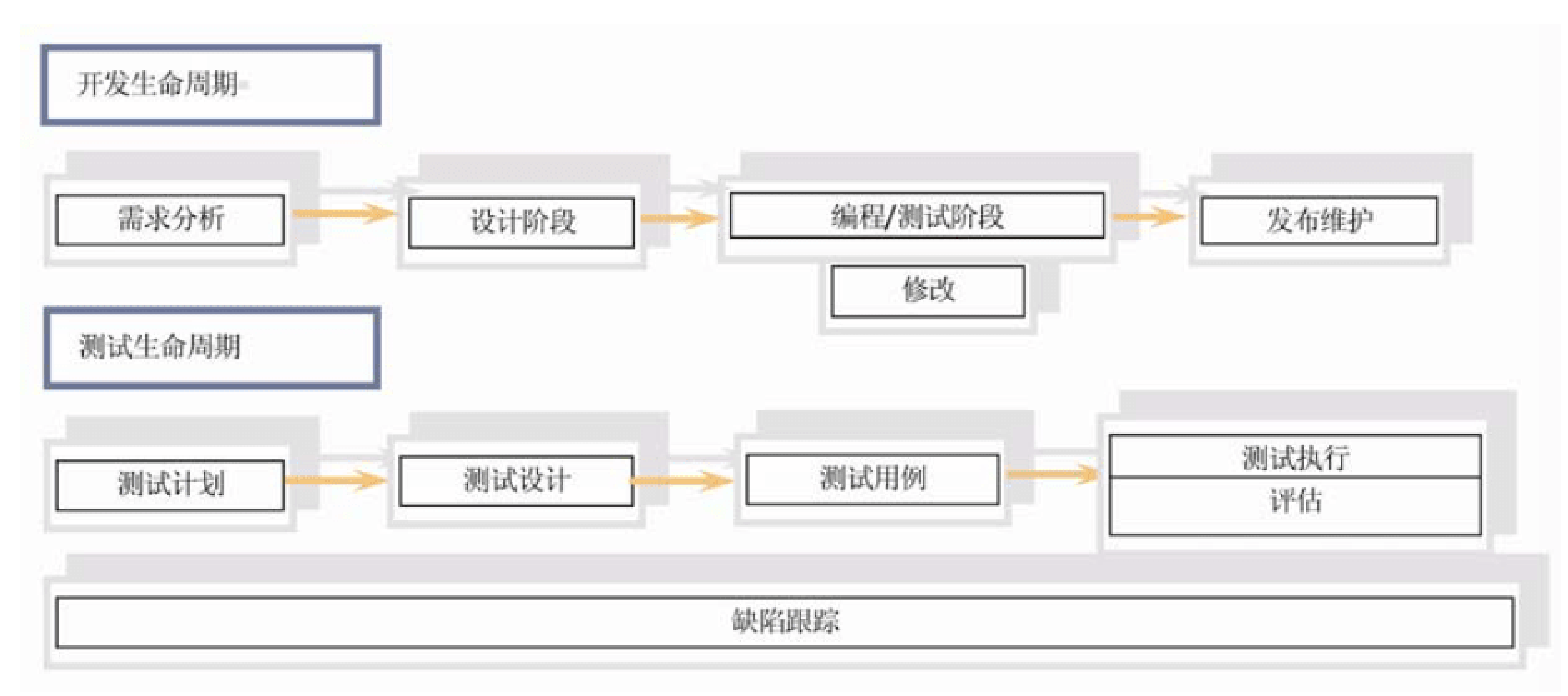


图 5-3　开发与测试生命周期

在缺陷生命周期中，缺陷的状态会发生变化。所有团队成员要对缺陷不同状态的概念有正确的理解和认识，才能避免对同一状态的不同理解和处理失当。从开始开发软件时就要统一设计好缺陷不同阶段状态的定义，比如：谁能修改一个缺陷的状态，修改的标准是什么等等。参见表 5-1。

表 5-1　常用缺陷状态说明表实例

序　号	状　态	描　述
1	New	测试过程中新提交的缺陷
2	Open	新提交并指派给开发人员处理的缺陷
3	Fixed	开发人员已修复的缺陷，等待测试人员验证
4	Reopen	缺陷修改未达到目标，重新指派给开发人员处理
5	Closed	缺陷已修复并已经通过测试人员验证
6	Rejected	开发拒绝的缺陷，不需要修复或者不是缺陷
7	Pending	当前版本不能修复的缺陷
8	Distract	在上线前仍未修复，后续跟踪的缺陷
9	Cancelled	被取消的缺陷

根据 IEEE Std 1044-1993 制定的缺陷生命周期中缺陷状态的案例如图 5-4 所示。

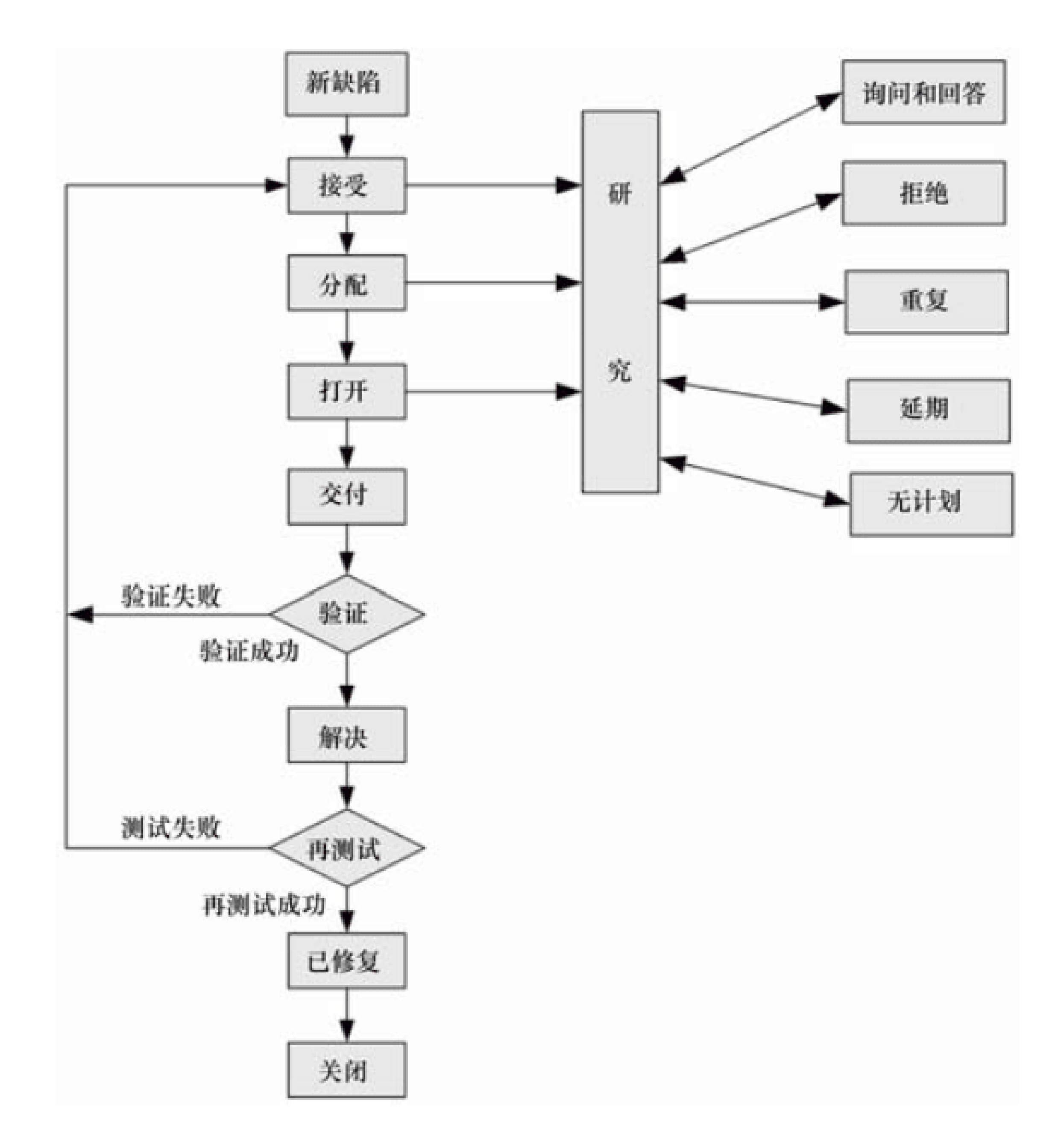


图 5-4　缺陷状态和缺陷生命周期

业界常用的缺陷生命周期是根据 IEEE Std 1044－1993 评估缺陷报告和缺陷分类而来的。和软件开发生命周期一样，缺陷也是由一系列的阶段和活动组成的，即缺陷同样具有生命周期。前面早已提到过缺陷生命周期，这里是另一个业界使用的示意图。它说明一个缺陷从开始发现、报告、解决、修复、验证等经不同角色人员采取不同处理行动，最后关闭一个缺陷的过程。从侧重于缺陷管理和缺陷状态变化的角度，并同时考虑到软件开发生命周期、测试生命周期和缺陷生命周期，我们来分析一下不同角色测试团队成员对于缺陷管理层次上的职责。如图 5-5 所示。

5.3.1　测试人员的职责

通常情况下，测试人员需要对该缺陷后续相关的状态负责，包括回答相关人员对这个缺陷信息的询问，以及在正式版本上进行再测试和回归测试。测试人员首先负责发现并在缺陷管理系统中详明地记录缺陷。当缺陷被修复后，测试人员负责验证缺陷是否被修复，并需要进一步测试是否引入了新缺陷。一般来说已解决的缺陷会指派给报告缺陷的测试人员，他们应该加上自己的意见和验证结果或者他们也同意项目经理关于缺陷不予修正的结论。当然分析测试人员要仔细分析缺陷不予修正理由，再慎重做出是否接受纠错的决定。最后，测试人员负责关闭缺陷。

测试人员可添加缺陷，可添加注释评论。他们可调整缺陷概要、缺陷级别、问题描述、附件附图、缺陷状态、测试版本、测试产品、功能模块、测试状态、问题定位、复测状态、注释评论。但缺陷复测人、复测日期和修改人由缺陷管理系统自动识别并生成。

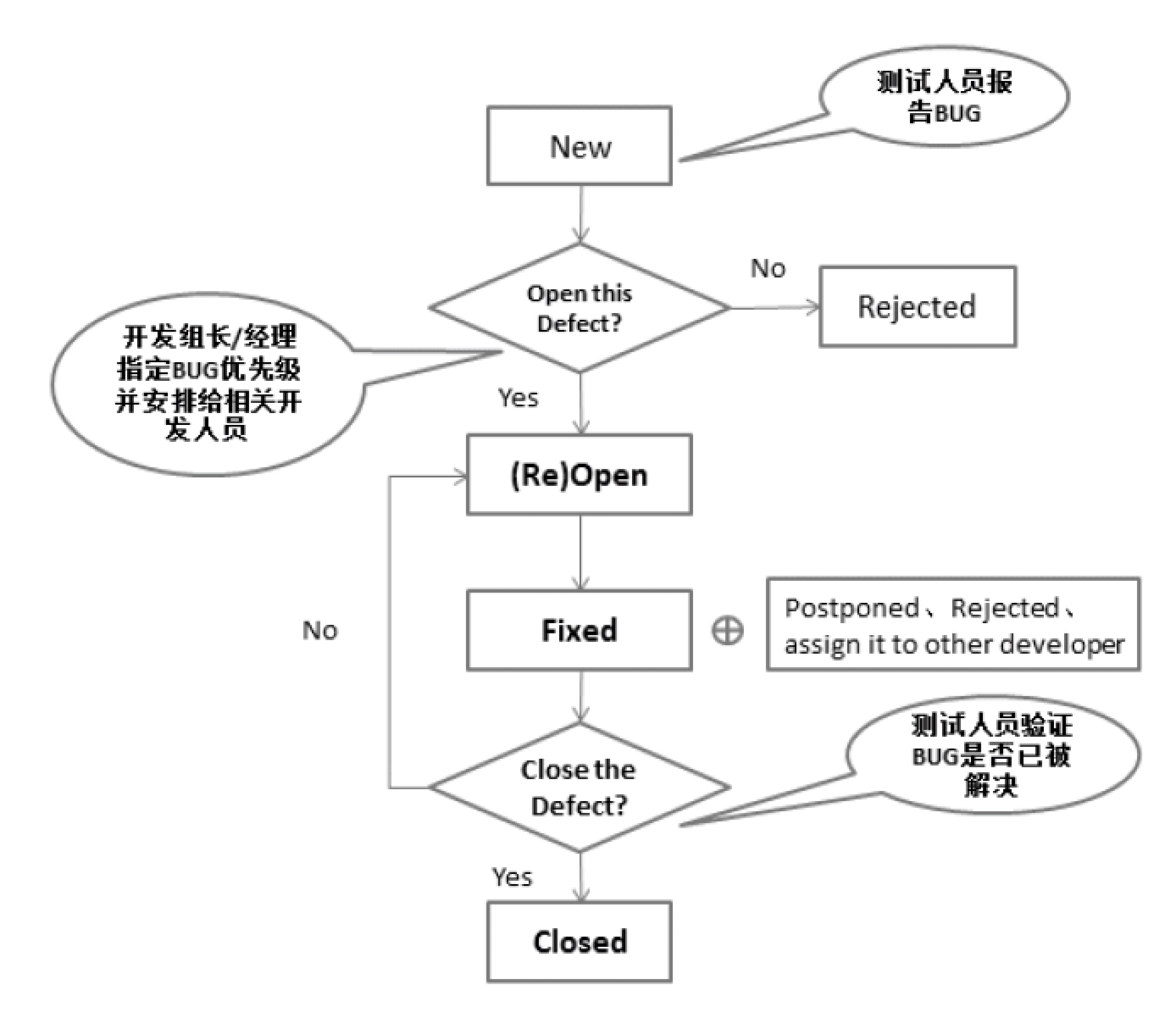


图 5-5　缺陷生命周期与主要参与角色

5.3.2　测试主管和测试经理的职责

测试主管和测试经理拥有全部管理、修改缺陷的权限。他们负责在高层指导和分配测试人员针对缺陷的工作。参与缺陷会审会议，对缺陷的解决方案从测试团队的角度提出意见与建议。比如，说明某个缺陷对用户或团队的影响，说服开发团队和项目经理尽快修复某个缺陷，而不是推迟或者不予修复。某些情况下，又要根据验证某些缺陷的代价说服开发团队和项目经理在本次迭代或者发布中推迟修复。功能模块提交后若开发人员发现缺陷，必须按照缺陷管理流程首先登记缺陷，严禁开发人员私自修改代码和私自进行代码优化之类的修改。

5.3.3　开发团队的职责

开发团队主要指对缺陷进行调查和修复的开发人员。注意在提交给测试人员正式再测试之前，需要对修改后的缺陷在开发环境进行验证。首先由开发经理或指定的开发部门技术负责人读懂并分析所有活动状态的缺陷，如果他认为应该修正这个缺陷，那他就把这个缺陷指派给相应的开发人员。不用修正的，就指派给项目经理。开发人员拿到缺陷就会尽量修正它，修正好后就把缺陷解决然后把缺陷指派到报告缺陷的人。

开发人员/需求人员可添加注释评论，可修改注释评论，可修改缺陷状态(不过无法直接标为 closed)、问题描述、处理意见、待测版本、修改人、修改日期，可添加缺陷。

开发组长/经理/需求经理的职责总结起来是除了开发人员的权限和权利范围，他们还可调整优先级、责任人(Assigned to)、缺陷概要等。

5.3.4　项目经理的职责

项目经理拿到缺陷后，要决定这个缺陷是否已经被别人报告过了，如果是，就要在缺陷

管理系统搜索，找到他认为是同样的缺陷，把后来报告的缺陷解决成“重复”再指派给报告缺陷的人。如果新的缺陷不需要修正，那项目经理应该从剩下三种 resolution(纠错决定)、By Design(原设计的行为)、Won't Fix(不用修正)和 Postpone(以后再考虑修正)中选一个适当的，再解决(resolve)缺陷并把解决的缺陷指派给测试人员或报告该缺陷的人。项目经理的另一个重要职责是在开发人员和测试人员有分歧时，站在用户和团队的角度，做出自己的判断，促进团队达成一致。

总之，项目经理可添加缺陷，可添加注释评论，可修改字段，可修改缺陷概要、问题描述、附件附图、缺陷状态(不过无法直接标为 closed)、修改人、优先级别、问题定位、处理意见、注释评论、是否重现、责任人、待测版本。

5.3.5　缺陷会审团队

业界对缺陷会审的英文常用词是“Triage”三方决策的意思。因为缺陷会审团队一般主要由开发团队、测试团队和项目经理共同组成。

1) 参加人员：程序员、程序设计人员、测试人员；

2) 目的：通过阅读、讨论与争论促使问题的暴露；

3) 活动：代码会审会。

缺陷评审委员会也可由更多角色人员参与。但主要是项目经理、测试经理、质量经理、开发经理以及资深的开发人员、测试人员等组成。他们对缺陷进行确认，并将其分配给相应的开发人员进行修复，同时对有争议的缺陷进行仲裁。当测试工程师发现并报告缺陷后，需要对缺陷的有效性和修复策略进行裁决，特别是有争议或者测试工程师和开发工程师无法单独做出判断时，需要缺陷会审团队介入。缺陷评审委员会有在缺陷管理系统里修改缺陷的全部权限。

总结起来，处理缺陷和改变缺陷状态的权限按主要角色总结如表 5-2 所示。

表 5-2　缺陷处理权限说明表

权限 / 角色	New	Open	Fixed	Reopen	Closed	Rejected	Pending	Distract
测试人员	√	√	×	√	√	√	×	×
开发人员	×	×	√	√	×	√	×	×
项目经理	×	×	√	√	×	√	√	√

5.4　缺陷报告管理

在前面的课本和实践中，我们已经了解了缺陷和缺陷报告。在软件测试过程中，对于发现的每个软件错误(缺陷)，都要记录该错误的特征和重现步骤等信息，以便相关人士分析和处理软件错误。为了便于管理测试发现的软件错误，通常要采用软件缺陷数据库，将每一个发现的错误输入到软件缺陷数据库中，软件缺陷数据库的每一条记录称为一个缺陷报告。

使用缺陷管理工具写缺陷报告的意义之一在于它可以准确、永久地记录软件缺陷和缺陷解决方案。某些情况下，通过电子邮件、即时通信软件甚至口头沟通都可以报告并解决缺陷，但用缺陷跟踪系统记录所有缺陷具有显著的优势。缺陷报告往往成为团队的组织过程资产，这些资料对开发未来版本的产品有用，对软件系统运行维护团队和产品支持部门都极具参考价值。这些部门依靠缺陷报告中的资料去了解产品，改进决策或帮助客户。

从测试工程师的角度，缺陷和测试用例是他们最重要的工作内容。测试用例描述测试的目的，缺陷则描述了这些测试用例的结果。软件测试工程师的工作成果之一就是他们发现的缺陷范围、深度、广度、全面性和质量，书写清晰明了的缺陷报告是一名实用型软件测试人才最基本的能力要求。本节特别介绍缺陷报告管理应注意的环节。

5.4.1　简明扼要的标题

缺陷的标题是缺陷报告中最显著的信息。参与软件研发的人员可以快速浏览标题去了解产品中或某一特定领域的缺陷类型。当用户去搜索某些缺陷时，也往往首先通过标题中的关键字进行查找，它使我们能便捷地找到具有某类特征的缺陷。标题还是产品功能或会审小组便于审查缺陷的重要信息，简洁扼要的标题便于参加缺陷会审会议的人员快速判定缺陷对用户或工程团队的影响，以制定最佳的修复策略。缺陷标题的字数没有严格限制，但是一般都只有短短的几十个字，不宜过长，需要提供准确的缺陷全面总结。它需要中肯的描述，提供恰到好处的信息。

1. 较好的标题实例

在某智能手机 App 的测试中，测试人员发现一个关于升级界面的缺陷，标题如下：“在升级程序时，状态栏中文字最右端有截断”。这个标题只有 17 个字，但是明确地说明了进行了何种操作、在什么位置出现了什么样的缺陷。软件研发团队中的开发工程师或者项目经理等干系人可以根据标题的描述初步判断缺陷的影响并不严重，不是 P1 级的缺陷，不必停下手中的工作立即修复。

2. 较差的标题实例

在某办公自动化软件测试过程中，有测试工程师报告一个缺陷，标题如下：“程序崩溃”。这个标题过于简单，没有说明何种情景下程序发生崩溃。团队中的开发工程师和项目经理等人员无法判断这个缺陷的影响程度。在前面讲述过，如果在常见情景下和非常见情景下程序崩溃类型缺陷的严重级是不同的，将无法判断是否需要立即修复。另一方面也没有描述缺陷发生在何处(与哪个功能相关)，也无法快速判断应由哪位开发工程师负责修复。

又如，在某办公自动化软件测试过程中，有测试工程师报告一个缺陷，标题如下：“打开各种应用程序，查看系统内存使用量，当系统内存使用达到 3G 以上时，启动 OA 软件，再打开一个大型报表，再打印这个报表，系统出错”。读者已经看出来这个是一个不太理想的缺陷标题，后经测试主管查看后，把标题修改如下：“系统在低内存情况下，打印一个大型报表，系统弹出对话框‘无法创建对象’”，并在报告中附上了测试中使用的报表。现在分析一下这个标题的问题。第一过于冗长，修改后的标题只有原来的大约一半，在绝大部分

缺陷跟踪管理系统中，都可以完整的显示标题，方便读者阅读。第二关键信息表达不清，比如没有描述系统物理内存有多少，如果硬件系统具有 8G 以上内存时，此缺陷有可能就无法重现；“系统出错”的描述令读者无法判断缺陷到底引发了何种问题，是崩溃了还是弹出了错误对话框，也就无法判断缺陷对系统的影响，无法快速判断缺陷的优先级，给缺陷的会审带来麻烦。

5.4.2　精确的问题描述

缺陷说明回答了标题中不便展开描述的细节，交代必要的背景，对缺陷进行精确详实的描述，以免歧义的发生。它包括对环境和情景的描述，软件版本对用户的影响，期望结果和实际结果；尽量使用数据说明问题，附上必要的附件，比如截屏、日志和转储文件等。文字尽量使用客观平实的语言描述客观事实，不要加入过多个人感情色彩。

仍以前面提到的办公自动化软件的缺陷为例。

1. 较好的缺陷说明

当系统物理内存耗尽时，办公自动化软件可以打开一个大型报表，但打印这个报表时，软件弹出报错对话框“无法创建对象”。系统物理内存充足时，无此问题。

系统物理内存耗尽是一种比较极限的情况，即使这样不能打印报表也是一种对用户影响较大的错误。另外，出错信息提示“无法创建对象”，对于最终用户来说过于技术化，不够友好。建议改为“当前系统负荷过重，请关闭部分应用程序后再试”等能给予用户操作一定指导的信息。

2. 较差的缺陷说明

“打开很多应用程序，当系统变慢时，办公自动化软件可以打开一个大型报表，但是死活就是不能打印这个报表，系统还弹出报错对话框‘无法创建对象’，让人完全看不懂神马意思，很不友好。”

这个描述语言不够严谨，比如“让人完全看不懂神马意思”这样网络化的语言还有“死活就是不能打印这个报表”这样带有强烈主观感情色彩的语言，会让读者感觉不是一名职业的测试工程师报告的缺陷而更像一名愤怒的用户。描述缺陷的时候尽量使用平实、不带主观感情色彩的语言，只描述所见的客观事实。作为一名测试工程师，更应从用户的角度出发，对如何修改缺陷提出自己的意见，可以参考较好的缺陷说明对出错信息的修改建议。

5.4.3　确认缺陷版本号

软件版本编号订定是指为软件设置版本号码的方式。通常，版本号码会以数字订定，但亦有不同的方式。

一般情况下，在缺陷报告中需要记录的版本号包括：测试工程师发现缺陷时所测试的版本号、开发工程师修复缺陷后生成的第一个版本的版本号以及测试工程师验证及关闭缺陷的版本号。

还有一些特殊情况，比如开发工程师的修复无效，需要将缺陷重新激活，需要记录重新

激活的版本号。有些不能稳定重现的缺陷，在修复后需要测试工程师根据一定策略连续在若干个版本上验证，如果没有重现这个缺陷，则需要记录所有进行过测试的版本的版本号。

5.4.4　简明的重现步骤

重现步骤往往包括在说明中，但有些系统单独列出这一缺陷报告的重要组成部分。重现步骤是指可以被缺陷工作流程参与者重新使缺陷出现的步骤。最令测试人员沮丧和倍感浪费时间的是开发工程师说，“这个缺陷并不能在我的电脑上重现。”一套好的重现步骤并不能保证这种情况不会发生，但会极大地降低这种可能性。另一种情况是当部分公司软件进入发布阶段后，打补丁的工作会交接给另外的维护团队，他们的一部分职责就是将之前解决方案“推迟”的缺陷修复，并发布补丁程序。随着全球合作进一步开展，维护团队可能与原始的研发团队不在同一处甚至不在同一时区工作。没有清晰的重现步骤，维护团队会不断向原始的开发团队询问，大大增加沟通成本，降低工作效率。

重现步骤也必须尽可能简明扼要。如果可能，要尽量简化重现步骤。虽然缺陷可能在 10 个步骤下被重现，依然建议花些时间看看是否有些不必要的步骤或者有更加简洁的重现方法。一方面使得缺陷更容易重现，减少争议；另一方面，精简的重现步骤会有助于迅速隔离缺陷起因，更方便地修复严重的缺陷，从而提高整个研发工作的效率。

在书写重现步骤时始终要采用换位思考，保证其他工程师可以稳定地重现所描述的缺陷。描述尽量简洁，使用简单句，避免复杂句。每一步骤只描述一个操作。使用客观语言描述操作步骤和客观事实，避免使用带有主观色彩的文字。

1. 较好的重现步骤

仍以之前章节中提到的办公自动化软件的缺陷为例。

1) 确认测试用计算机正确连接并设置打印机；

2) 打开办公自动化软件；

3) 打开多个应用软件或者使用相关工具占用大量系统内存，使系统处于低内存状态。标志：在任务管理器中看到系统可用物理内存已经耗尽或者极低。这时操作系统会明显变慢；

4) 单击主界面上“导入报告”按钮；

5) 导入附件 test.xls 文件；

6) 导入完毕后，单击主界面上“打印报告”按钮。

2. 较差的重现步骤一

1) 打开办公自动化软件；

2) 打开若干个大 Word 文档；

3) 再打印一个大报告。

与较好的重现步骤比，这组重现步骤过于简略，关键的细节信息有所缺失。(1)没有确认打印机是否已经正确连接并设置好；(2)没有明确指出要重现这个缺陷，系统应处于物理内存耗尽状态；(3)没有给出打印报告的具体步骤。这些信息的缺失，严重影响到其他工程师尤其是开发工程师重现这个缺陷，开发工程师不得不付出额外的精力去尝试和猜测，甚至通过电

话、邮件或者其他方式与报告缺陷的测试工程师沟通，增加整体工作负荷，增加不必要的沟通开销。

3. 较差的重现步骤二

1) 确认测试用计算机正确地连接到打印机；
2) 打开打印机，在测试计算机上安装打印机驱动程序；
3) 打印测试页，确定打印机已经可以正确使用；
4) 单击开始菜单，找到办公自动化软件图标；
5) 单击办公自动化软件图标；
6) 启动办公自动化软件；
7) 打开新的 IE，访问某门户网站；
8) 查看系统物理内存使用情况；
9) 重复步骤 10 直到系统物理内存占用超过 98%；
10) 检查系统是否已经变慢；
11) 在办公自动化软件主界面上单击“导入报告”按钮；
12) 导入附件 test.xls 文件；
13) 导入完毕后，单击主界面上“打印报告”按钮。

这组重现步骤的问题是过于繁琐，没有突出重点。大量篇幅用在如何设置打印机以及如何消耗系统物理内存上，而且消耗系统物理内存一段也没有精确的指出这些步骤的目的是消耗系统物理内存。书写报告和阅读报告重现缺陷都需要花费大量不必要的时间。可以参考较好的重现步骤中的描述，对重现步骤进行精简。这些不足往往是由于书写测试报告的测试工程师对系统功能不熟悉，对计算机系统技术缺乏深入了解造成的，常见于刚刚从事测试工作的初级测试工程师，这也是初级测试工程师通往高级测试工程师必经的道路，出现类似问题的学者不必过分担心，可以通过阅读学习优秀缺陷报告并不断实践来逐步提高自己书写缺陷报告的能力。

5.4.5　正确使用严重级和优先级

前面教学已多次解释和谈到缺陷报告中的严重级和优先级。实际工作中所有公司和团队都会定义和服从预先设定的严重级和优先级。但有时也发生一些认识不同和有争议的情况。以下是几个实例。

1. 开发对于问题处理的争议

有些软件公司会以问题处理的优先级的分级法直接取代问题严重性的分级法。这就等于由项目经理或所授权的人士决定处理问题的优先级。这种局面并非没有可能，其中一个可能就是团队本来是从很精简的人手开始慢慢壮大，而因为以前人手精简，当时的严重性和优先级的决定权很自然都会同时落在发现问题的人手上，也就是没有区分。所以，团队越大，越有必要清楚两种分级方法。

以优先级取代严重性这种单方面的游戏规则就是为什么测试人员和开发人员经常在问

题的处理方式上出现争议的原因。

2. 缺陷和需求变更界定出现含糊所惹来的争议

也有些软件公司在发现问题的过程中，发现一些设计上的问题，也列入系统缺陷来处理。例如界面本身虽然能正常运作，但由于设计时对于某方面的应用没有考虑在内，因而导致在某应用需求无法满足。这也是一个辨认系统缺陷的一个误区，而 QA 在这个盲点上进行处理必然会惹来争议。尤其对于开发来说，发现设计上的问题，从编程和开发的角度看绝对不是一个缺陷。

以上的例子似乎说明了产品的当前状况是完全符合产品当前的设计的。其实，发现了这种所谓的问题，是对当前产品可改善的建议，理应界定为需求变更，可以向负责产品设计的部门提出。当建议被接纳继而开发后，就是套用 QA 的日常工作流程，对此新改良的设计再进行测试，再去评估问题的严重性，在问题追踪系统内跟进。

所以对于以上情况，发现的问题虽然同样可以有其严重性和优先处理等级，但所发现的在类别上有根性的区别：它不是缺陷，而是需求变更。

3. 问题分级要有明确具体的指标，切忌问题分级制度形同虚设

虽然有些团队会利用一些问题追踪系统，系统内或有高、中、低，或有 1 至 5 之类的分级，然而，没有明确指标和具体含意的分级是形同虚设的，甚至会影响整体进度，令团队发生处理问题上的争议。这些指标需要开会讨论，或者在团队内发布消息。

5.4.6　管理缺陷严重性和优先级

正确区分和处理缺陷严重程度和优先级是软件质量保证的重要环节。因此，正确处理和区分缺陷的严重程度和优先级是所有软件开发和测试相关人员的重要职责，需要正确理解缺陷严重程度和优先级的含义，同时认识到这是保证软件质量的重要环节，应该引起足够的重视。将比较轻微的缺陷设置成高严重程度和高优先级的缺陷、夸大缺陷的严重程度将影响软件质量的正确评估，耗费开发人员辨别和处理缺陷的时间；而将严重的缺陷报告成低严重程度和优先级的缺陷，这样会掩盖许多严重的缺陷。如果在项目或者软件发布前，发现还有很多由于不正确分配优先级造成的严重缺陷，将需要投入很多人力和时间进行修改，影响软件的正常发布；或者这些严重的缺陷成为漏网之鱼，随着软件一起发布出去会影响软件的质量和降低用户使用软件的信心。

1. 优先级(Priority)与严重性(Severity)的区别

从根本上，优先级是从项目管理和时间管理的观点来厘定高低的，而严重性是从质量管理的观点来思考的。

处理问题的严重性，以质量作为出发点，应针对所产生的问题是否会对产品造成严重的缺陷来决定等级；其决定权，通常掌握在 QA 人员手中。

处理问题的优先级，以整体项目的进度、质量、市场以及需求所造成的影响作为出发点，

决定应对的措施；其决定权，可以分散至负责交货或者售后服务的部门，而最终决定权通常掌握在项目经理手中。

2. 优先级(Priority)与严重性(Severity)的分级

缺陷管理也包括对已发现的缺陷的统计分析以及对缺陷的预防。从不同的角度描述缺陷的影响，可以将缺陷按严重级和优先级进行分级(如表 5-3 所示)。

表 5-3　缺陷优先级

序　号	优　先　级	描　　述	平 均 时 间
1	1-低	暂时不影响继续测试；可在方便时解决	3 天
2	2-中	部分功能无法继续测试；需要优先解决	2 天
3	3-高	测试暂停，无法进行；必须立即解决	1 天

缺陷的严重级是指因缺陷引起的故障对软件产品和用户的影响程度。一般以数字表示缺陷的严重级，数字越小级别越高。以下是一种分级方式：

1 级：Critical　系统不能执行正常工作功能或重要功能失效，甚至造成用户数据丢失，危及用户人身安全。

2 级：Major　严重地影响系统正常功能或基本功能失效，且没有替代办法。

3 级：Minor　系统正常功能受影响，但存在可行的替代办法。

4 级：Cosmetic　使操作者不方便或遇到麻烦，但它不影响执行工作功能或重要功能。

举例：1 级：Critical 级缺陷

1962 年美国的水手 1 号由于程序员在输入方程式时的一个笔误(¯ versus -)造成任务失败，史称“The most expensive hyphen in history”。

举例：2 级：Major 级缺陷

某型智能手机操作系统计算器内部十进制/二进制转换过程中丢失精度造成计算结果有误，如图 5-6 所示。智能手机的 CPU 与传统计算机的 CPU 一样，只能处理二进制数据，在进行十进制数字运算的情景下，要先转换成二进制再把二进制的计算结果再换回十进制。在转换过程中，由于二进制数据位数限制，会造成精度损失。

举例：3 级：Minor 级缺陷

某型中文手机输入法，默认设置会在每行输入字符后加入一个空格，当用户在 IE 的密码框输入密码时，密码最后也会自动加入一个空格，造成用户无法正常登录网站。修改默认设置后该缺陷被修复。

举例：4 级：Cosmetic 级缺陷

某型手机采用的字体部分字形方向显示错误，如图 5-7 所示。

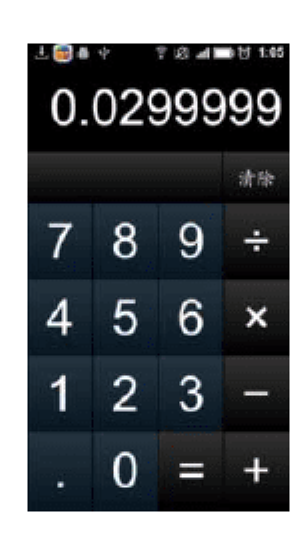
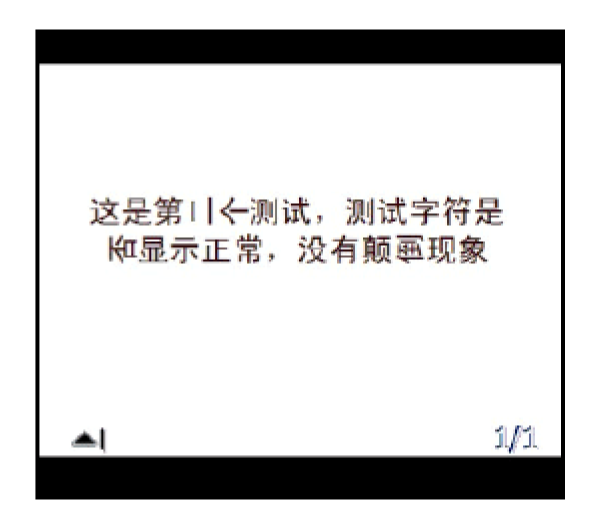



图 5-6　手机操作系统计算器缺陷


图 5-7　某型手机字体部分缺陷

缺陷的优先级是指综合考虑各种因素后，工程团队修正缺陷的顺序，极端情况就是否修正缺陷。一般以数字表示缺陷的优先级，数字越小级别越高。以下是一种分级方式。

1 级：必须修正(Must Fix)缺陷对用户或者工程团队有极大影响，必须被立即解决，一般负责修复缺陷的工程师应该停下手中的其他工作，在 24 小时内修复。

2 级：应修正(Should Fix) 缺陷应在本次迭代(里程碑或冲刺)或者本次发布结束前被修复。

3 级：有时间就修正(Fix if Time)，这是一个小缺陷，可推迟时间取决于产品开发的阶段。

缺陷的优先级的甄别取决于很多因素，一般来说与严重级成正比关系，即严重级高的缺陷其优先级也高，被优先修复。但某些情况下，优先级并不是如此简单，需要缺陷会审(Triage)会议来决定优先级。

还有一种定义缺陷分类的方法：

- A 类——严重错误，包括：

(1) 由于程序所引起的死机，非法退出

(2) 死循环

(3) 导致数据库发生死锁

(4) 数据通讯错误

(5) 严重的数值计算错误

- B 类——较严重错误，包括：

(1) 功能不符

(2) 数据流错误

(3) 程序接口错误

(4) 轻微的数值计算错误

- C 类——一般性错误，包括：

(1) 界面错误(详细文档)

(2) 打印内容、格式错误

(3) 简单的输入限制未放在前台进行控制

(4) 删除操作未给出提示

- D 类——较小错误，包括：

(1) 辅助说明描述不清楚

(2) 显示格式不规范

(3) 长时间操作未给出用户进度提示

(4) 提示窗口文字未采用行业术语
(5) 可输入区域和只读区域没有明显的区分标志
(6) 系统处理未优化

• E 类——测试建议(非缺陷)

5.4.7　及时更新缺陷状态

每个缺陷从创建以后就会指派(Assign To)给某位工程师或者项目经理，这些人被称为缺陷的负责人(Owner)。缺陷处于不同状态时会指派给不同的负责人，这些负责人要及时更新缺陷的状态。最常见的情况是，当缺陷被创建并明确是由哪位开发工程师修复以及修复的时间期限时，这名开发工程师要在规定的修复时间期限修复缺陷并及时将缺陷状态更新为修复，并指派给一名测试工程师，通常是缺陷的发现者，来验证修复是否有效。如果遇到困难不能及时修复，也要将遇到的困难详细地记录到缺陷报告中，并及时通知相关工程师寻求帮助。当缺陷被开发工程师修复后，会被指派给测试工程师进行验证，一般是指派给发现这个缺陷的测试工程师。一旦有了包含缺陷修复的版本可用，这名测试工程师要尽快验证并更新缺陷状态，如果验证修复有效则需关闭缺陷，如果修复无效需重新激活缺陷并重新指派给相应的开发工程师。在某些情况需要对缺陷进行会审，一般会指派给项目经理，然后进行会审，最后决定指派给哪名工程师。

不同状态英文翻译的叫法和环节显示有所不同，但基本要点都一致。图 5-8 是某个缺陷状态转换图实例。

5.4.8　测试人员跟踪缺陷

测试团队的不同成员负责从不同角度密切跟踪缺陷是常见的实践。测试工程师主要负责跟踪某个特定缺陷的创建、验证和关闭等状态转换。测试主管和测试经理负责整体跟踪不同状态的缺陷的数量变化、趋势等。跟踪过程中特别注意检查缺陷报告质量。测试人员执行，测试组长/经理把关，以开发人员的角度来审查缺陷报告中缺陷的描述，看其是否描述清了问题，不便描述的把工程文件或截图作为附件提交。具体可从以下几方面考虑。

(1) 缺陷报告总体质量：分类是否准确、叙述简洁、步骤清楚、有实例、易再现、复杂问题有据可查；

(2) 问题描述时有无具体信息：模块或功能点=>测试步骤=>期望结果=>实际结果=>其他信息，可依据实际情况调整；

(3) 单一性：尽量一个报告只针对一个软件缺陷，报告形式应方便阅读。在主报告之后应注明不同的条件；

(4) 简洁性：每个步骤的描述应尽可能简洁明了。只解释事实、演示和描述软件缺陷必要的细节，不写无关信息；

(5) 易重现性：问题必须在自己机器上能重现方可在缺陷管理系统中报告(个别严重问题重现不了也可报告，但需标明)；

(6) 复杂的问题处理：应附截图补充说明或直接通知指定的修改人；考虑到网络数据传

输效率，截图的文件格式建议用 JPG 或 GIF，不建议用 BMP；

(7) 报告使用语言：不允许使用抽象词句：比如“有错误”之类。

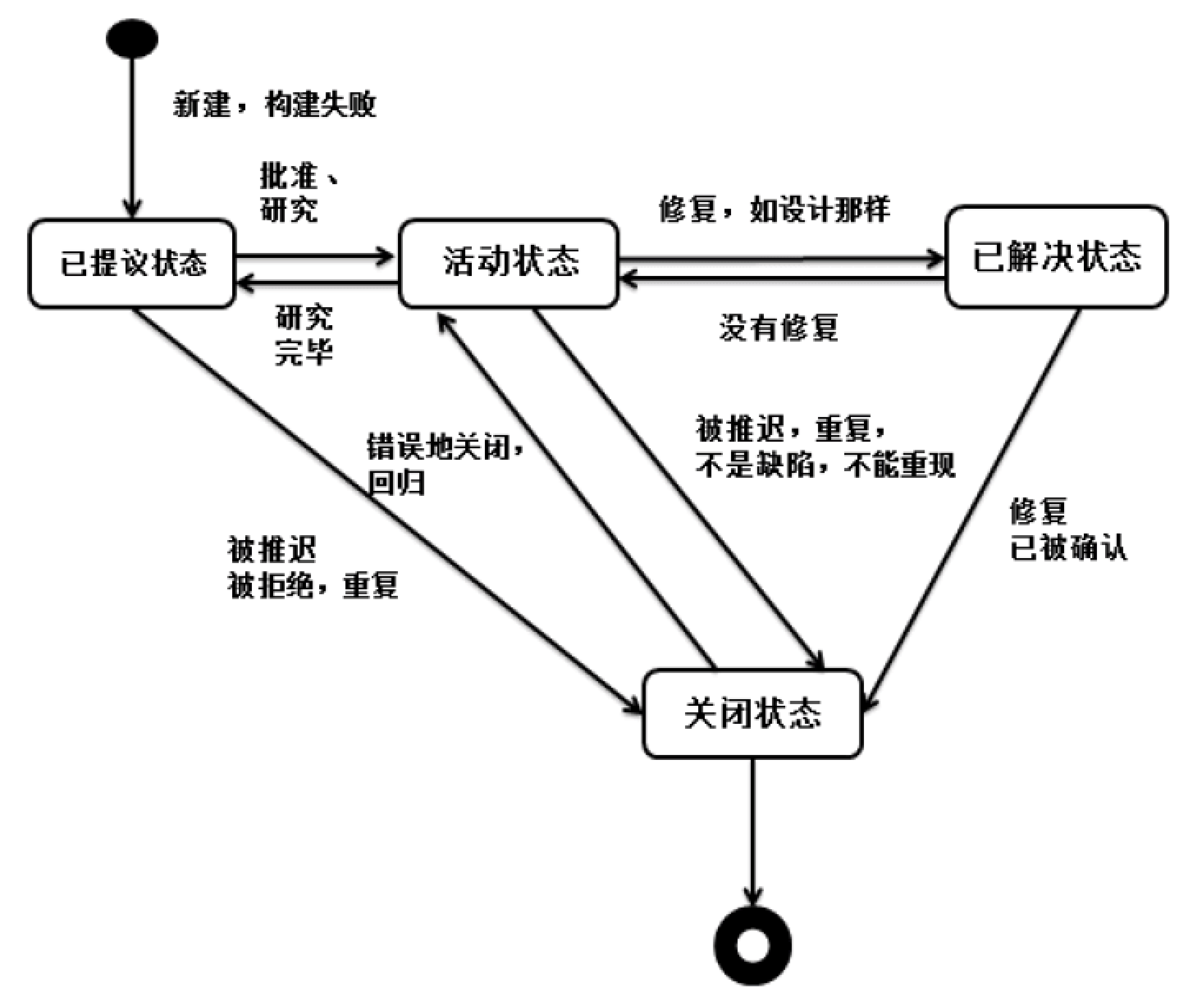


图 5-8　缺陷状态转换图实例

5.4.9　缺陷的发现时间

缺陷发现的时间是很重要的，必须准确。因此缺陷管理系统都设计了自动记录报告缺陷的时间。缺陷报告中还需要详细记录每次缺陷状态更新的时间，包括创建、修复、验证、关闭、重新激活等。在基于数据库的缺陷跟踪管理系统中，系统通常会自动记录这些时间点。这些数据经过处理后可以帮助管理团队对团队能力进行评估进而正确地对未来的工作预期时间进行估算。比如缺陷的平均修复时间、已修复缺陷的平均验证时间。从整个软件项目的生命周期看，哪个时间段内发现缺陷的数量最多，哪个时间段内修复缺陷的数量最多等等。

习题与思考题

1. 请简述缺陷管理的目的？
2. 缺陷有哪些状态和缺陷状态之间的关系(需要通过哪些软件实现状态的转换)？
3. 请描述在缺陷管理中测试人员的职责？
4. 缺陷会审团队由哪些成员组成？缺陷会审的意义？
5. 在缺陷报告书写中，好的重现步骤有哪些特点？
6. 当测试工程师和开发工程师对缺陷的有效性有争议时，怎么办？
7. 请简述为了写好一个缺陷报告应注意哪些事项？

第6章　软件测试流程管理

软件测试是融合进整个软件开发生命周期的一个完整过程。软件测试相关的流程有很多，不同类型的软件公司和团队都可能制定适应各自实际需要的测试流程。测试的尽早介入是软件测试的一个基本原则；不应把软件测试仅看做运行软件待测产品相关的检查活动或者软件开发的一个阶段。为有效实现软件测试各个层面的测试目标，需要和软件开发过程一样，定义一个正式而完整的软件测试过程，即涉及各个软件测试活动、技术、文档等内容的过程，用以指导和管理软件测试的各项活动和提高测试效率和测试质量，同时改进软件开发过程和测试过程。本章将从不同方面解释测试流程的基本概念和实践，并介绍常见的测试相关流程图以供参考。

6.1　软件测试流程管理基础

任何事情都有一个循序渐进的过程，从计划到执行再到实现。软件流程就是按照这种思维来定义我们的开发过程，它根据不同的产品特点和以往的成功经验，定义了从需求到最终产品交付的一整套流程。流程告诉我们该怎么一步步去实现产品，可能会有那些风险，如何去避免风险等等。由于流程来源于成功的经验，因此，按照流程进行开发可以使得我们少走弯路，并有效提高产品质量，提高用户的满意度。本节侧重测试流程的相关定义，说明测试流程的流程图以及理解和管理好测试流程的意义。

6.1.1　流程图

流程图是流经一个系统的信息流、观点流或部件流的图形代表。也是一种有效地指导测试时间的工具。过程流程图是利用一定的符号将实际的流程以图形方式呈现出来，以便于确定可能的变量。通过流程图，可以对要改进的过程有一个全面的、统一的了解；帮助项目团队确定过程中一切可控与不可控的变量以及可能出现的缺陷。图 6-1 的测试策划流程图就是一个针对测试不同环节的一个简单测试流程图。它可以帮助项目团队所有人员以及管理人员了解需要考虑的测试环节和相应的顺序。图 6-1 直观地告诉读者要考虑的几个步骤和具体内容，以及顺序。

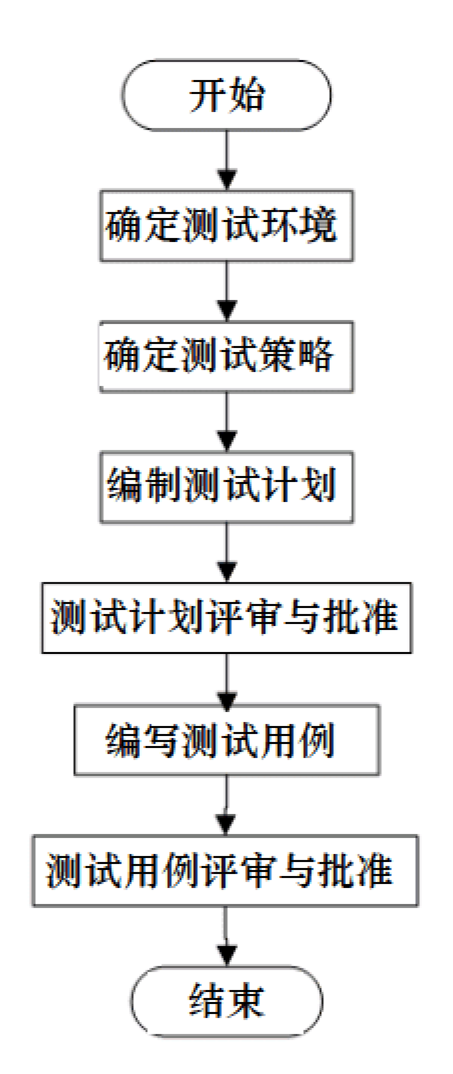


图 6-1　测试策划流程图

6.1.2　测试流程管理的意义

流程的重要性不言而喻，任何一个软件产品都由众多担任不同角色的软件工程师协作完成，想象一下，如果每个人都按自己的想法随心所欲、天马行空的做事情，没有统一的指挥调度，没有经过协商全体认可的测试规范，整个软件产品项目的开发必然一团糟。

实施流程管理，对软件开发项目团队、部门以及整个企业至少有以下重要意义：

1) 角色分工的统一和集中分配便于管理和绩效考核。

由于有明确的做测试具体事务的流程，就可以监督测试人员是否按照流程办事和是否提交预期的、高质量的测试结果。也便于在进行绩效考核时作为基本要求来评估。

2) 沟通所需的软件开发和测试流程环节和结果、步骤帮助团队成员明确各自的工作任务。

具体的测试流程作为指南性参照在团队中可以讨论和沟通，帮助团队成员明确工作要求和彼此帮助解决遇到的问题。

3) 明确测试流程便于领导层及时发现隐患并采取行动

由于整个团队都遵守和参照统一的流程，测试负责人和其他项目负责人就可以监督和检查测试工作结果。如发现问题及时纠正，并根据情况采取行动。

4) 便于新员工快速学习应做的工作，并融入团队工作

测试流程常用文字、流程图或其他方式清晰明了，便于新员工掌握和遵守。当然这也使他们能尽快融入团队并满足团队对他们的基本要求。

如上所述，实施测试流程管理对企业测试工作管理的作用是多方面的。用一句话来说，那就是测试流程管理让测试工作变得更简单和规范化。

通过实施测试流程管理，让所有的测试人员明确工作分别由谁做、怎么做以及如何做好。由于职责清楚、责任分明，将大大提高测试团队的敏捷程度和竞争能力。测试流程管理有助于员工潜能的释放和积极性的发挥，也将大大增强企业的核心竞争力，从而达到测试团队运行有序、效率提高的目的。

6.2　软件测试的一般流程

软件测试活动是软件研发中一个有机组成部分，软件测试流程也符合一般软件项目的特点，主要分为计划与设计阶段、实施测试阶段和总结阶段。本节着重介绍测试流程，设计具体的交付物(比如测试计划、测试用例等的书写等)将在其他章节中详细介绍。不同软件产品测试的流程都会有所不同。本节介绍几种典型的测试相关流程，并用流程图等直观方式表示。

测试流程常可以用文字描述。比如以下的几个例子。

1) 实例一：文字描述流程：项目立项后开始写测试计划，根据需求编写测试需求，根据测试需求编写测试用例，根据测试用例执行测试，没通过的测试用例提交测试缺陷报告，再进行回归测试，直到测试结束编写测试总结，每个步骤都需要审核通过。

2) 实例二：简单文字和字符表示顺序：测试申请——提交缺陷——解决缺陷——验证缺陷——关闭缺陷——生成测试报告。

3) 实例三：详细文字描述：

测试流程包括的环节是：测试需求分析，测试计划编写，测试用例编写，执行测试，缺陷记录，回归测试，判断测试结束和测试报告提交。测试流程各环节说明如下：

(1) 需求：阅读需求，理解需求，与客户、开发、架构多方交流，深入了解需求。此环节由测试经理或测试团队负责人完成。

(2) 测试计划：根据需求估算测试所需资源(人力、设备等)、所需时间、功能点划分、如何合理分配安排资源等。此环节由测试经理或测试团队负责人完成。

(3) 用例设计：根据测试计划、任务分配、功能点划分，设计合理的测试用例。此环节由测试主管、有经验的测试工程师完成。

(4) 执行测试：根据测试用例的详细步骤，执行测试用例。由初级测试人员负责。

(5) 执行结果记录和缺陷记录：对每个测试用例记录测试结果，有缺陷的在测试管理工具中编写缺陷记录。

(6) 缺陷追踪：追踪缺陷的负责人分配给你追踪的缺陷。直到缺陷得到修复。此步骤由所有测试人员完成。

(7) 测试报告：通过不断测试、追踪，直到被测软件达到测试需求要求，没有重大缺陷。

(8) 用户体验、软件发布等。

6.2.1　开发模式与软件测试流程

作为广义的软件测试，ISTQB(International Software Testing Qualifications Board)定义了一个完整的软件测试流程，将测试相关的所有活动都纳入到了其中。图 6-2 是 ISTQB 定义的软件测试流程逻辑框图。

图 6-2 可以看出，软件测试流程由 5 个阶段组成。其描述的软件测试流程好像各个阶段是顺序进行的。但实际上有些测试阶段在时间上是可以有重叠甚至是并行进行的，例如：测试分析和设计、测试实现和执行阶段在时间上可能是有重叠的，而测试控制活动会贯穿于整个项目。

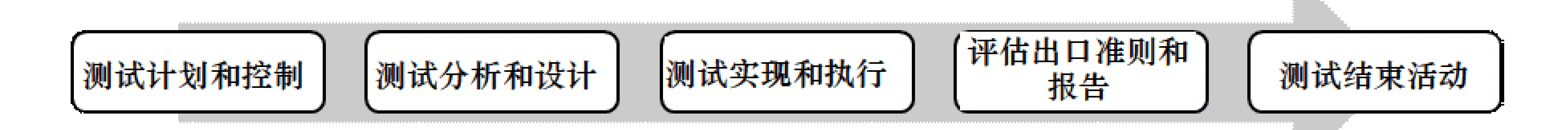


图 6-2　ISTQB 定义的软件测试流程

开发模型

不同的软件团队会根据情况采用不同的开发模型，其中的测试流程也要根据不同的开发模型而有所不同。我们已了解到常见的开发模型有瀑布模型、螺旋模型和迭代模型。不同流程模型适用于不同类型的项目。在这些不同的开发模型中，软件测试的流程也有所差异。由于敏捷方法的特性，将在专门的小节中讨论。以下我们做一下说明和对比。

1) 瀑布模型

瀑布模型(Waterfall Model)代表一个软件开发架构，于 1970 年被温斯顿·罗伊斯(Winston Royce)提出。其核心思想是按工序将问题化简，将功能的实现与设计分开，便于分工协作，即

采用结构化的分析与设计方法将逻辑实现与物理实现分开。瀑布模型代表的开发过程是通过设计一系列阶段顺序展开的，从系统需求分析开始直到产品发布和维护，每个阶段都会产生循环反馈，因此，如果有信息未被覆盖或者发现了问题，那么最好“返回”上一个阶段并进行适当修改，项目开发进程从一个阶段“流动”到下一个阶段，这也是瀑布模型名称的由来。瀑布模型也是其他开发模型的基础。软件开发和测试在瀑布模型下的流程有很多关联和交叉。

在开发软件前，首先要花费大部分时间对该软件做好详细的调查评估，选择合适的突破点，选择合适的开发语言和开发工具，制定详细的分步骤测试开发计划。

一旦开发与测试计划制定结束，将不会有大的更改。在过程中应完全按照计划进行，无法返回起点。这也是为什么称此模型为瀑布的原因，瀑布是流下去无法返回的单向流程。

在每个步骤即将完成时，都会对这一步骤进行总结，如果进行下一步骤的条件不具备，将停留在原地，等待条件成熟。瀑布模型看起来给人很专业的感觉，所以，对软件开发人员有比较高的要求。图 6-3 是瀑布模型流程图，图 6-4 是对应瀑布模型的一种实用的简单测试流程。

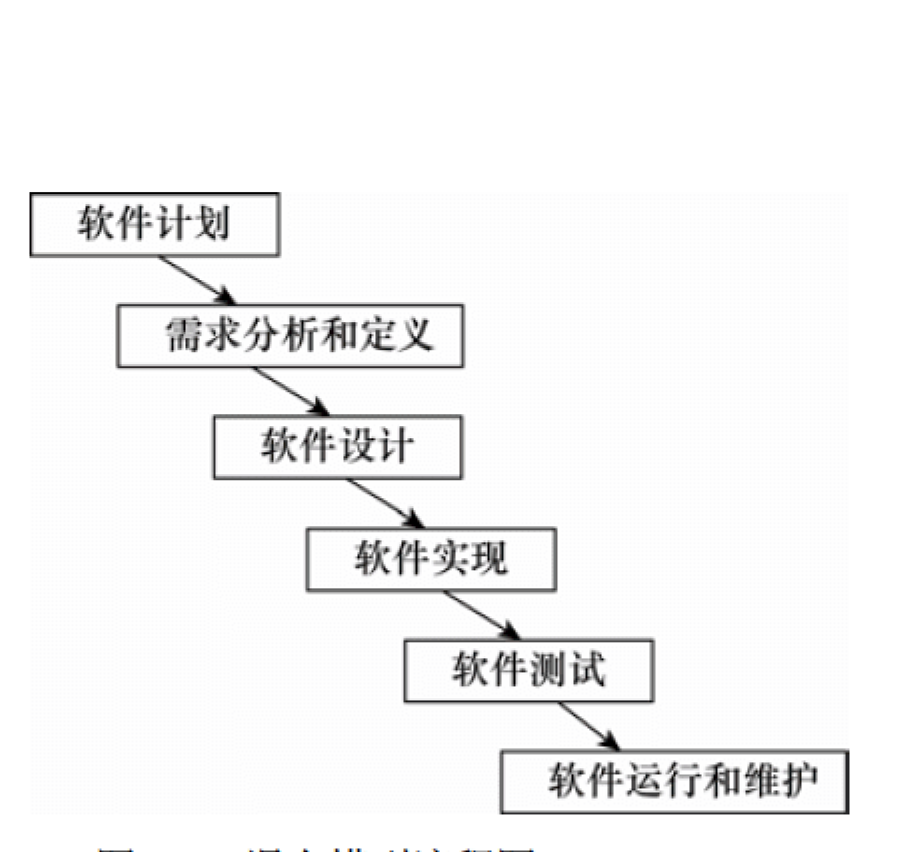


图 6-3　瀑布模型流程图

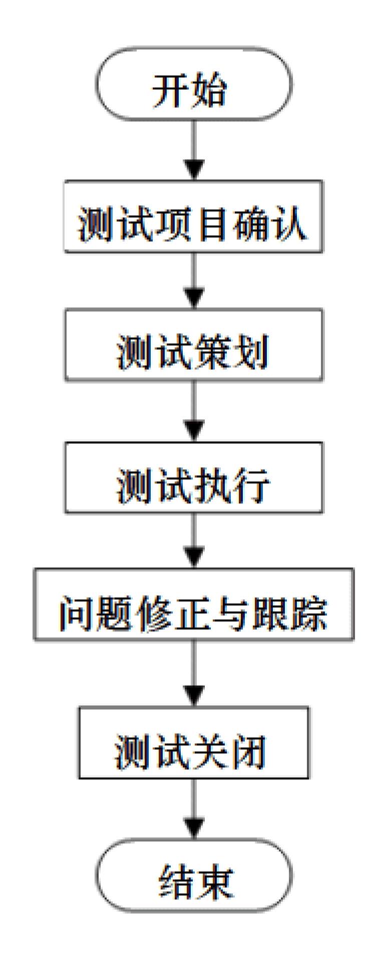


图 6-4　瀑布模型测试流程图

如图 6-4 所示，相对瀑布式的测试流程主要包括确认测试项目的需求、范围、目标等，然后是策划，也就是测试计划制定。进而是执行测试，发现软件缺陷并报告问题，修正后要验证修正并跟踪问题，到最后结束测试。

瀑布开发流程要对开发的软件有细致、全面、准确的了解。如果理解错误，可能导致计划失败，难有重来的机会。软件开发人员要具备坚定执行计划的能力。这种流程会期望测试是十分全面的，而且通常会有多于一轮的完整测试，因为在测试初期会有大量缺陷被发现，我们需要根据功能和优先级对测试用例进行分组并分批测试，以保证测试进度不会因严重缺陷而停滞不前。

正是由于瀑布模型有如此要求，因而产生了瀑布模型的缺点：那就是无法完美适应当今要求快速开发产品，从而占领市场的软件行业现状。因为制定详细的、完整的计划很难，聚合很多专业的开发人员有时候也很难，而市场对于软件更新换代的要求期限越来越短。为了

适应变化，后期人们又提出了“螺旋模型”。

2) 螺旋模型

螺旋模型(Spiral Model)采用一种周期性的方法来进行系统开发流程。该模型以进化的开发方式为中心，在每个项目阶段使用瀑布模型法。这种模型的每一个周期都包括需求定义、风险分析、工程实现和评审 4 个阶段，由这 4 个阶段进行迭代。软件开发过程每迭代一次，软件开发又前进一个层次。图 6-5 是螺旋模型流程图。

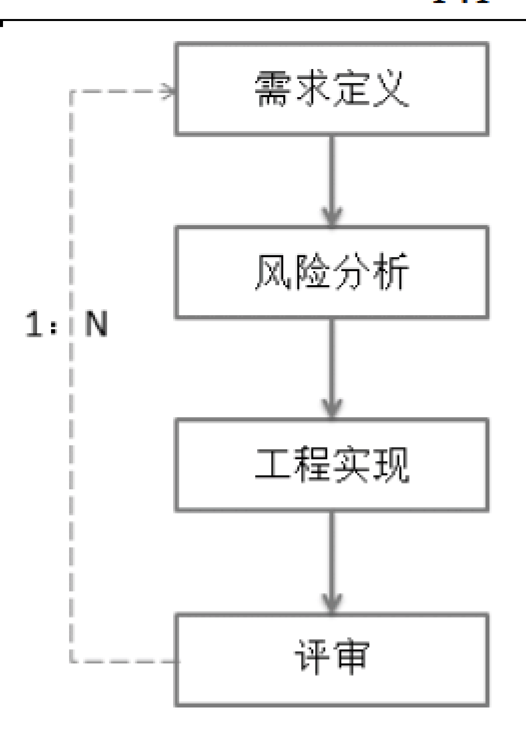


图 6-5　螺旋模型流程图

螺旋模型基本做法可以理解成它是在“瀑布模型”的每一个开发阶段前引入一个风险识别、风险分析和风险控制，它把软件项目分解成一个个小项目。每个小项目都标识一个或多个主要风险，直到所有的主要风险因素都被确定。图 6-6 是更细化的、体现测试环节的螺旋模型流程图。

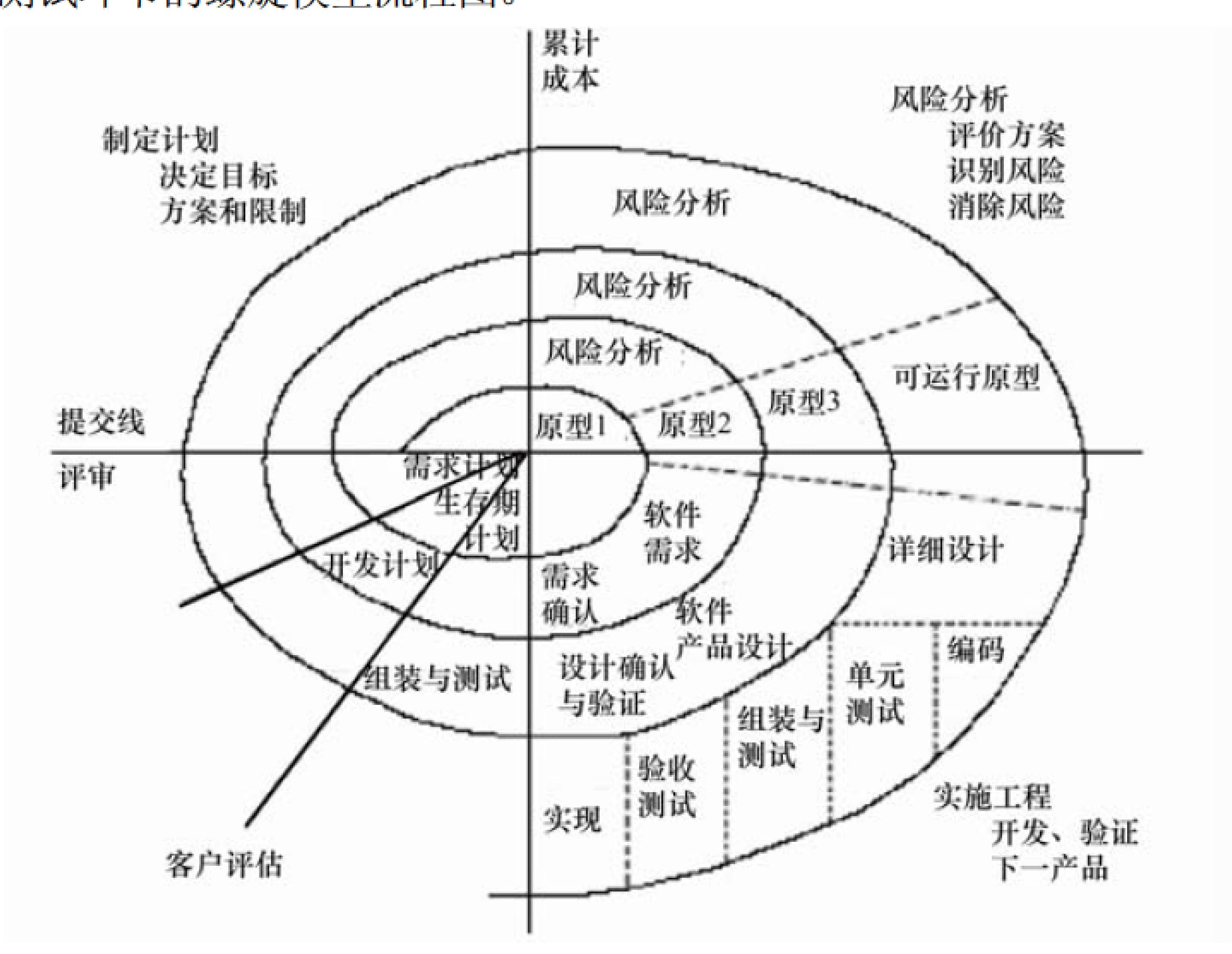


图 6-6　细化的螺旋模型流程图

螺旋模型沿着螺线进行若干次迭代，图中的四个：限代表了以下活动。

(1) 制定计划：确定软件目标、选定实施方案、弄清项目开发的限制条件；

(2) 风险分析：分析评估所选方案，考虑如何识别和消除风险；

(3) 实施工程：实施软件开发、测试和验证；测试环节包括设计、编码、单元测试、组装与测试，验收测试，实现等。

(4) 客户评估：评价开发工作、提出修正建议并制定下一步计划。

螺旋模型由风险驱动，强调可选方案和约束条件从而支持软件的重用，有助于将软件质量作为特殊目标融入产品开发中。

除瀑布式和螺旋式开发模型外，还有迭代模型(Iterative Model)。

3) 迭代模型

迭代模型也常被描述为“分段模型(Stagewise Model)”，是 RUP(Rational Unified Process，统一软件开发过程)推荐的周期模型。常见的定义为：迭代包括产生产品发布(稳定、可执行的产品版本)的全部开发活动和要使用该发布必需的所有其他外围元素。在某种程度上，开发迭代是一次完整地经过所有工作流程的过程：需求、分析设计、实施和测试工作流程。如图 6-7 所示。实质上，它类似小型的瀑布式项目。RUP 认为所有阶段都可以细分为迭代。每次迭代都会产生一个可以发布的产品，这个产品是最终产品的一个子集。在迭代模型中测试的流程可以参考瀑布模型中的测试流程图。

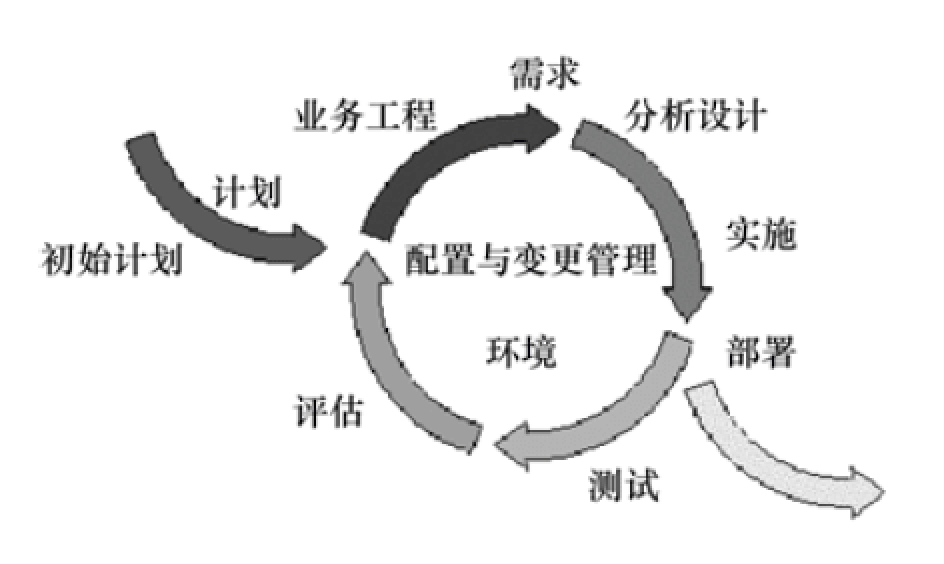


图 6-7　迭代模型项目周期流程图

图 6-8 是比较详细的测试环节和流程图。下面逐一进行解释和说明。

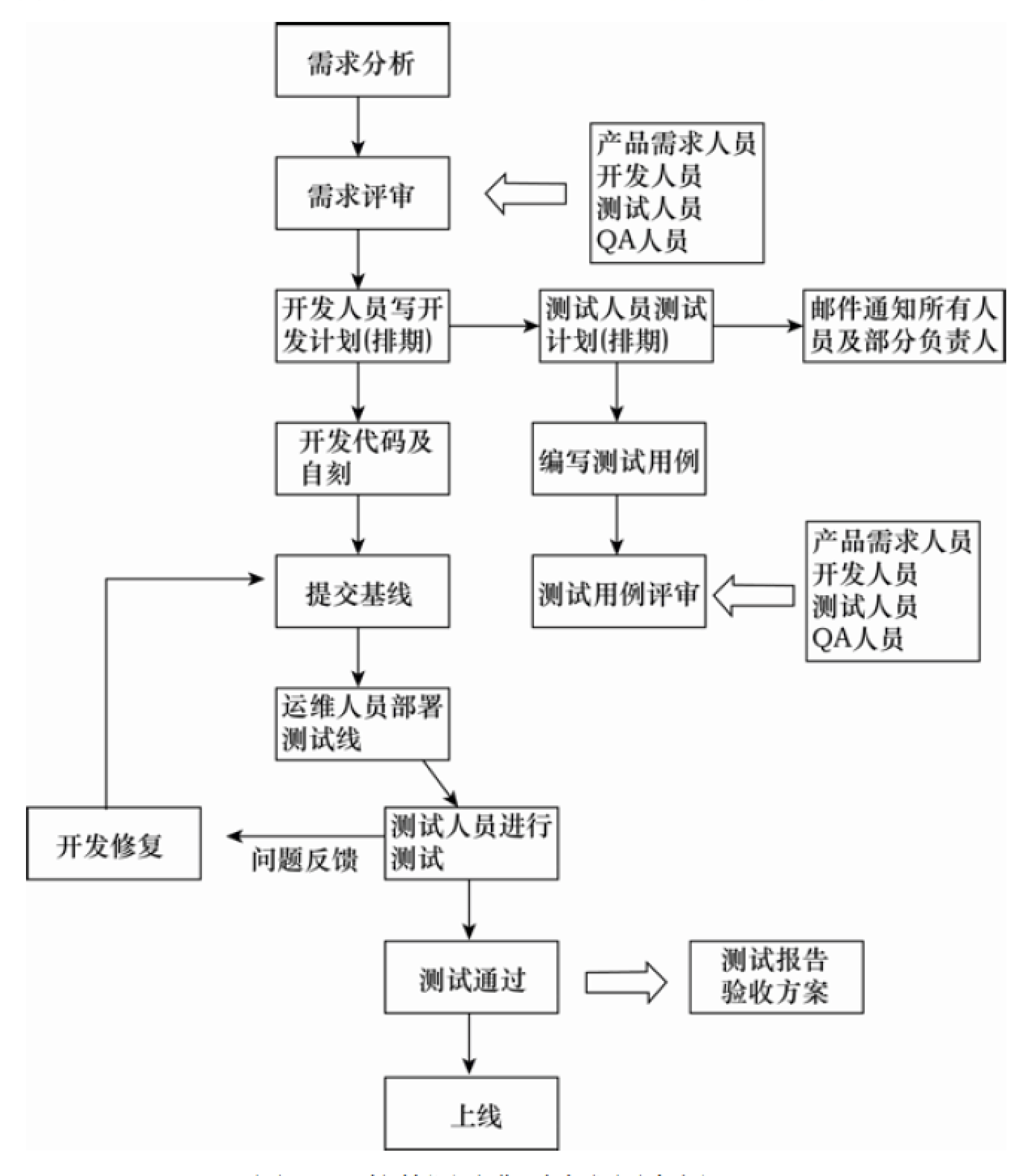


图 6-8　软件测试典型流程图实例

1) 需求分析

需求分析由产品人员制定，他们要做的不是一份简单的文档，而是细化每一个功能的细节，每一个按钮的位置，对于稍大或复杂一点的需求都进行建模。

2) 需求评审

这个阶段所有参与项目人员都会参与：开发人员、测试人员、QA 人员。测试人员会根据需求和功能的可测试性考虑和审视，开发人员则需要考虑功能实现的方案与可行性。测试人员主要是对需求的理解提出疑问，以便能根据需求写用例。QA 人员是最终对软件质量进行验证的人，所以会从项目整体方面了解。

3) 开发人员编写排期

开发人员根据需求功能点和技术设计出开发计划(排期)。

4) 测试计划排期

测试人员根据开发计划和测试计划给出详细测试排期，包括具体测试时间，也就是开发功能完成后的时间进行几轮测试、测试种类、回归测试整合测试等。然后，项目经理，开发组长，测试组长要一起审阅项目的开发与测试计划，所有项目干系人都同意后，再发送给各部门负责人及参与项目的所有人员。

5) 编写测试用例

根据详细的需求文档和测试计划开始进行用例的编写。

6) 用例评审

在用例进行评审前，建议相关评审人员首先阅读测试用例和相关需求设计文档，以便事先了解用例对哪些功能进行验证以及验证的细节。

然后，测试人员组织进行用例评审，在评审期间，参与评审的项目干系人对用例提出意见，例如：与实际功能不符合的有哪些，产品人员会通过用例对功能的具体实现进行把握等等。用例评审是一个让所有项目干系人对项目各方面需求和整体了解的好机会。通常功能设计的一些缺陷会在这个时候被发现出来，而这个阶段的缺陷修正成本相对在测试期间发现要低很多。

7) 提交基线

开发人员完成所有功能后，会对自己的功能进行一个自测。自测完成后提交给测试人员进行测试。测试人员执行功能测试(或者自动化测试)，发现的问题通过缺陷管理工具进行反馈，开发人员对问题进行修复，然后准备第二轮测试。

测试人员完成第一轮测试后，需要写测试报告，交付给所有项目相关人员。然后对基线后的第二轮进行测试，第二轮会对第一轮中发现的问题进行重点回归。

8) 测试结束

通常按测试计划会规定几轮测试，并制定测试结束标准。完成测试后，测试人员根据制定的标准提供测试报告。

验收方案是交由 QA 进行验证的。有些公司的流程中将测试与 QA 分开，测试人员重点关注的是功能是否可以正常运行。QA 关注的是整个流程的质量以及最终用户的质量。有些公司 QA 与测试是不区分的，但这对测试的要求会更高，除了关心功能，还需要关心整体流程与质量。

图 6-8 显示的开发和测试流程都属于比较规范和普遍的，测试人员融入整个流程能有效

地提高软件产品的整体质量。但这个项目流程要求比较多的各种文档，比如测试计划、测试用例、测试结论、测试报告、验收方案、问题的提交跟踪。这都要求时间和其他资源，不适合敏捷开发流程。

业界还有很多专家认为一个规范化的软件测试过程可以综合总结为包含 4 种基本的测试活动：一是拟定软件测试计划、方案；二是设计和生成测试用例、准备测试数据；三是执行测试，记录原始数据，对缺陷进行管理；四是生成软件测试报告、缺陷的统计和报表。

图 6-9 是另一个比较全面考虑测试活动的测试主要步骤流程图。该流程图还同时考虑了“是”与“否”的不同情况下的步骤。其中的环节在之前都解释过，在此不再赘述。

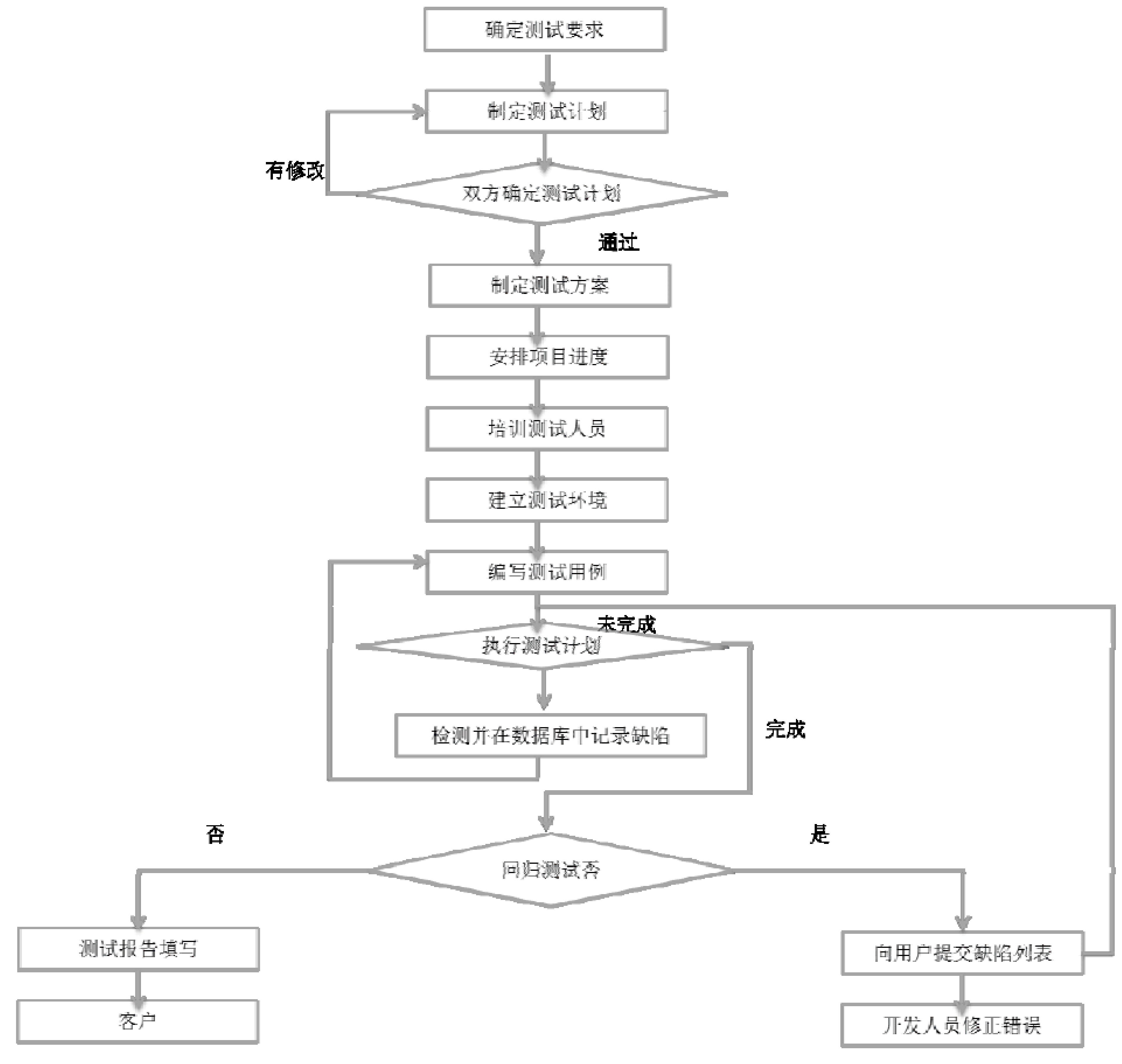


图 6-9　测试主要步骤流程图

6.2.2　计划与设计阶段

在测试计划与设计阶段，测试经理和测试团队开始介入软件产品的研发计划。测试经理通常通过和开发经理和项目经理开会的形式获得项目启动的相关信息。对项目的可测试性、测试范围、测试策略、测试方法、产品质量要求和时间人力等资源进行预估，并和测试团队成员一起讨论并完成测试计划。同时开发经理和项目经理要完成开发计划，测试团队和开发

团队分别对开发计划和测试计划进行审阅(Review)并签字确认。

在软件测试计划和设计阶段应完成测试计划和测试用例设计等文档，作为本阶段的交付物和下一阶段的指导性文件。《ANSI/IEEE 软件测试文档标准 829-1983》将测试计划定义为："一个叙述了预定的测试活动的范围、途径、资源及进度安排的文档。它确认了测试项、被测特征、测试任务、人员安排以及任何偶发事件的风险。"软件测试计划是指导测试过程的纲领性文件，包含了产品概述、测试策略、测试方法、测试区域、测试配置、测试周期、测试资源、测试交流、风险分析等内容。借助软件测试计划，参与测试的项目成员，尤其是测试管理人员，可以明确测试任务和测试方法，保持测试实施过程的顺畅沟通，跟踪和控制测试进度，应对测试过程中的各种变更。

在测试计划与设计阶段，有几个具体的测试环节与流程需要专门在这里加以分析，如下。

1. 测试计划和测试设计

测试计划和测试设计都是前期测试人员要介入的重要环节。有了需求说明书后，就该开始测试计划的测试设计，如图 6-10 所示。测试设计侧重的是将测试计划阶段制订的测试需求分解、确定测试方法和策略、细化为若干个可执行的测试过程，并为每个测试过程选择适当的测试用例。测试计划是要根据用户需求报告中关于功能要求和性能指标等规格说明书，准备相应的测试需求报告，比如制订黑盒测试的预定标准，目的在于使所有测试工作都将围绕测试需求进行；同时，还要选择适当的测试内容，合理安排测试人员、测试时间及测试资源等。

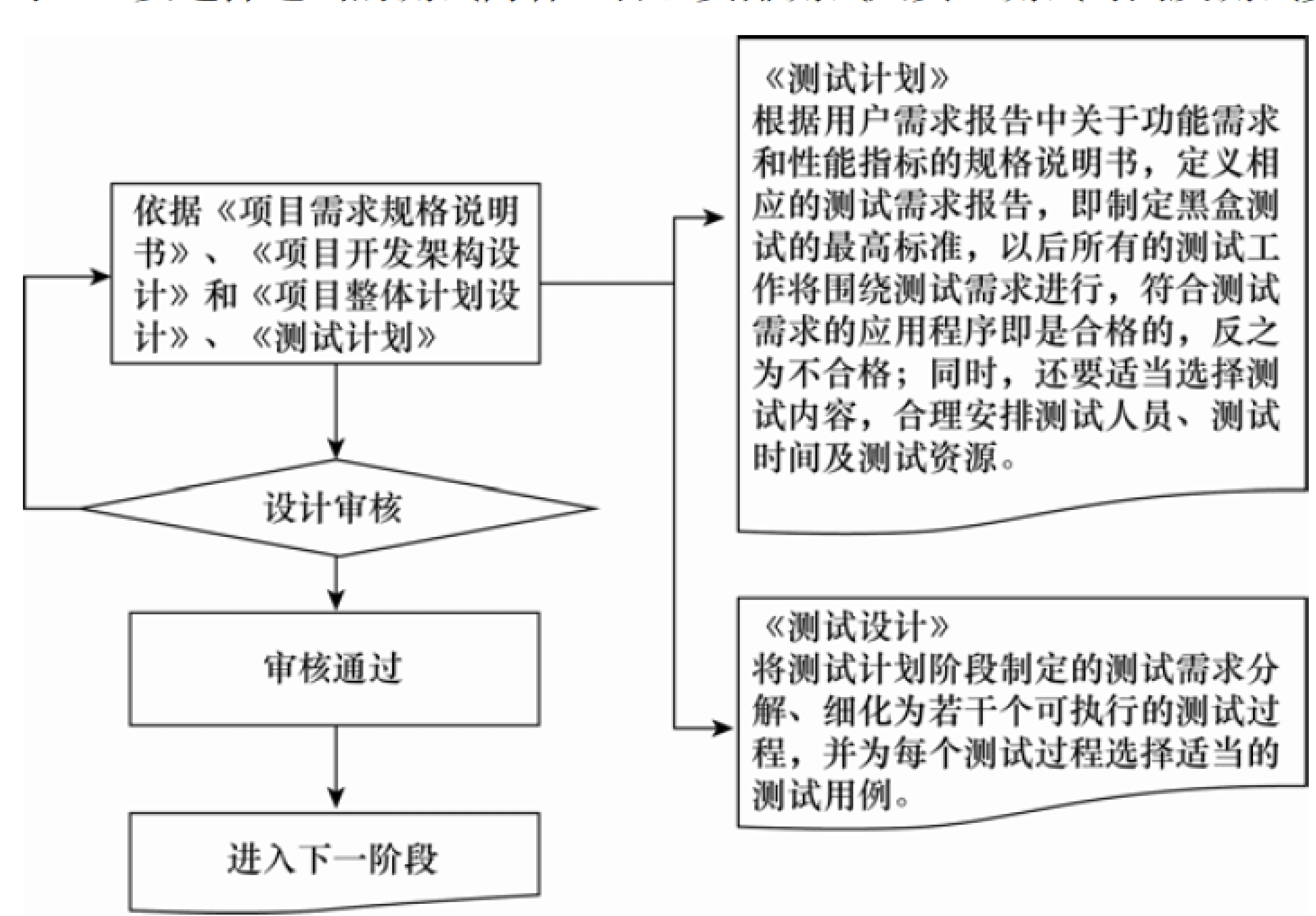


图 6-10　测试计划和设计流程图

2. 测试项目确认

在有了产品功能说明书、测试需求等信息后，确认测试项目，接下来也要评估和分析几个环节。测试项目确认也可以用简单流程图说明。图 6-11 就是一个实例。

在测试计划确定后，测试团队就可以着手开始设计测试用例。测试用例的设计要从不同角度展开。测试用例设计通常由测试主管指导测试工程师完成。

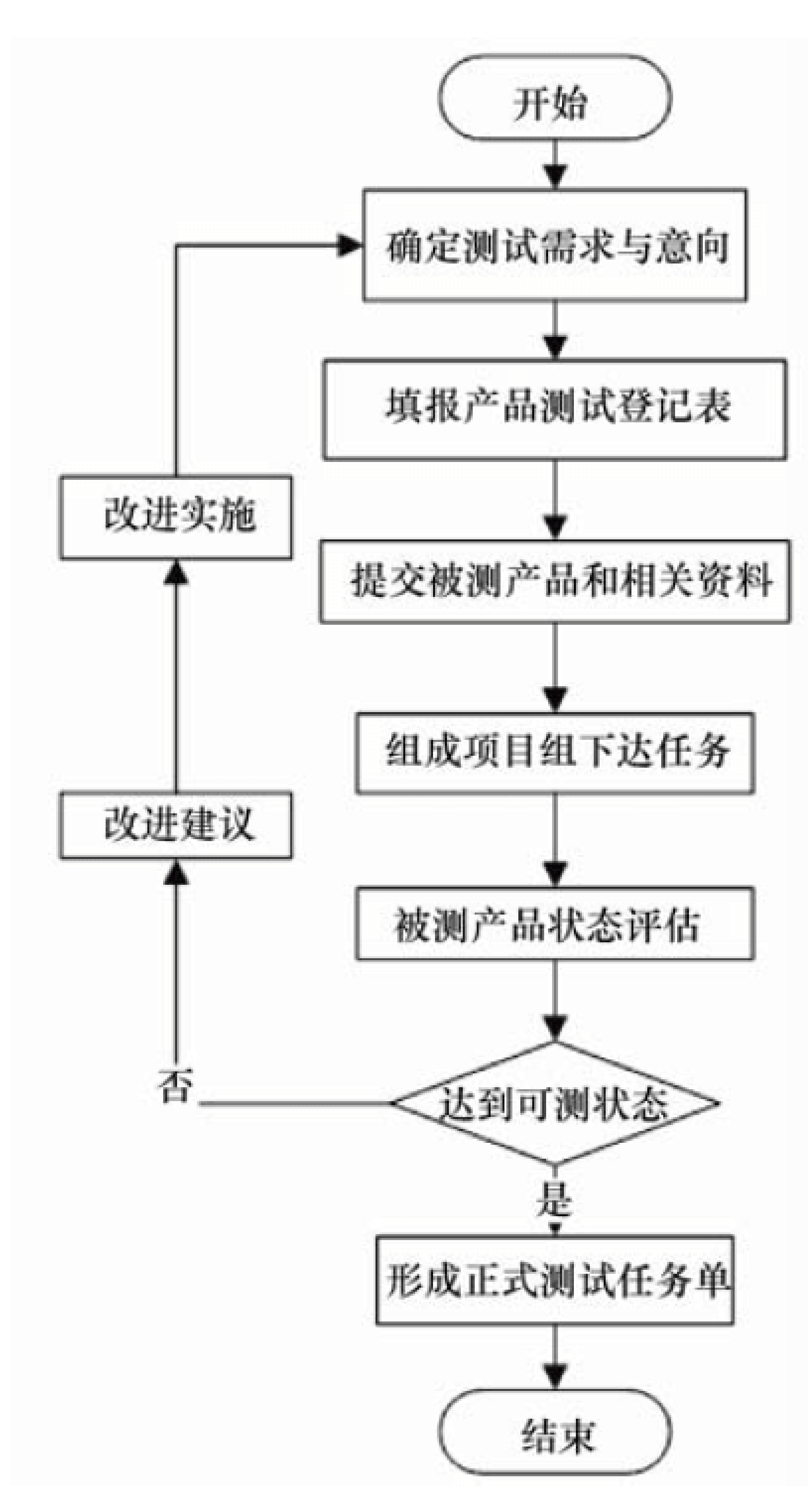


图 6-11　测试计划和设计流程图

3. 计划与设计阶段详细流程实

图 6-12 体现了计划与设计阶段流程与不同阶段的任务。体现了测试人员从一开始立项阶段就参与计划的具体任务，一直到设计评审。设计阶段有不少重要文档需要评审。

此流程图说明的做法和环节适用于瀑布式软件研发模式或迭代式研发模式前期设计和计划。敏捷开发模式的特点之一是不用写很多测试文档而且需要快速推进研发进度，因此很多测试是由研发人员自己完成的。不同软件研发项目需要根据需求和资源等决定那种流程更适合。

6.2.3　实施测试阶段

软件测试的执行也就是实施测试的阶段。一般包括 4 个常见环节：(1)执行测试用例；(2)记录原始测试数据；(3)记录和报告缺陷；(4)对所发现的缺陷进行跟踪、管理和监控。

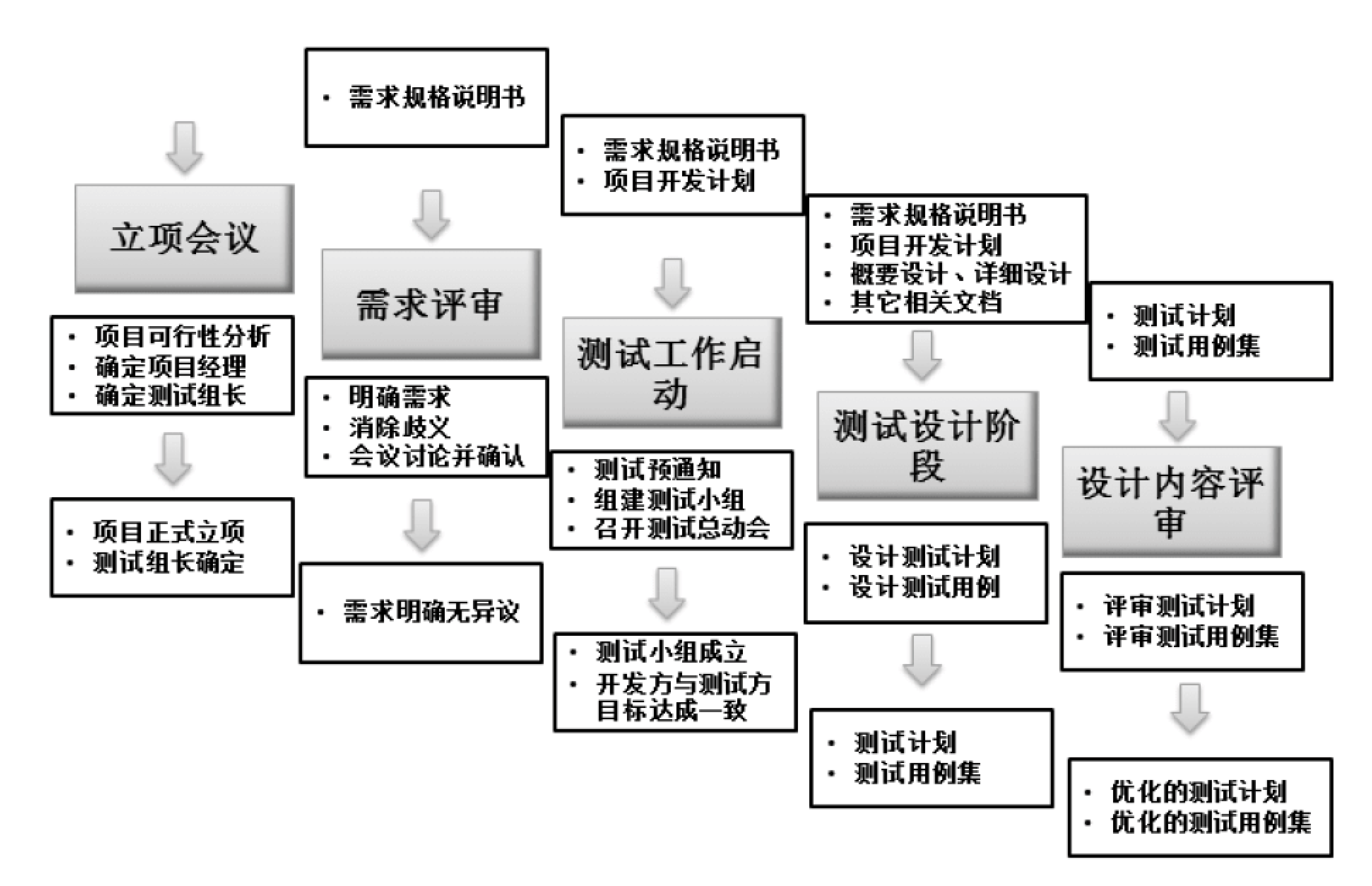


图 6-12　计划与设计阶段详细流程实例

测试的实施阶段主要活动是开发和执行测试用例。开发和执行测试用例需要根据测试计划的规定有计划有步骤的执行。这个阶段测试活动的工作量最大，沟通成本最大，发现的缺陷最多。

自动化测试工程师根据测试计划，领取分配给自己的自动化测试用例开发工作，按照测试计划中规定的时间完成自动化测试用例的开发工作，以便手动测试工程师执行测试用例。测试用例可以按照优先级进行分类。高优先级的测试用例一般覆盖最常见场景，确认最常用功能可用；低优先级的测试用例通常覆盖不常见场景或者比较极端的情况。优先级高的测试用例执行频度高，优先级低的测试用例执行频度低。高优先级的测试用例每天甚至每次代码做导入(Check-In)都被作为回归测试而执行以确保软件产品最基本的功能可用。而较低优先级的测试用例则可能每周或者每次迭代才被执行一次。具体执行频度是在测试计划和设计阶段制定并写入测试计划的。除测试计划内要完成的测试内容外，根据项目的进展，也有可能临时对测试计划进行调整，临时增减测试活动。测试经理或测试主管要与开发团队及项目经理密切合作，高质量地完成测试工作。

实施测试阶段另一个重要的工作是提交测试报告。每次根据测试计划执行完一次测试活动后，都应提交完整准确的测试报告，测试报告主要包含内容：软件产品的版本号、测试人员和时间、被执行的测试用例、测试结果、新发现的缺陷、验证被修复的缺陷的结果。

在实施测试阶段，具体的测试都要按照要求和流程进行。比如系统测试就有推荐的流程。明确流程就是具体为实施测试的人员提出需要考虑的因素和步骤。下面列举几个具体测试流程加以说明。

1. 系统测试

系统测试是将已经确认的软件、计算机硬件、外设、网络等其他元素结合在一起进行信息系统的各种组装测试和确认测试。图 6-12 就是一个系统测试的流程图。

图 6-13 显示了系统测试需要考虑的主要环节：缺陷报告、新测试版本，回归测试等步骤和相互顺序的关系。系统测试流程帮助测试人员按照流程规定将经过集成测试的软件作为系统计算机的一个部分，与系统中其他部分结合起来，在实际运行环境中对计算机系统进行一系列严格有效的测试，以发现软件潜在的问题，保证系统正常运行。

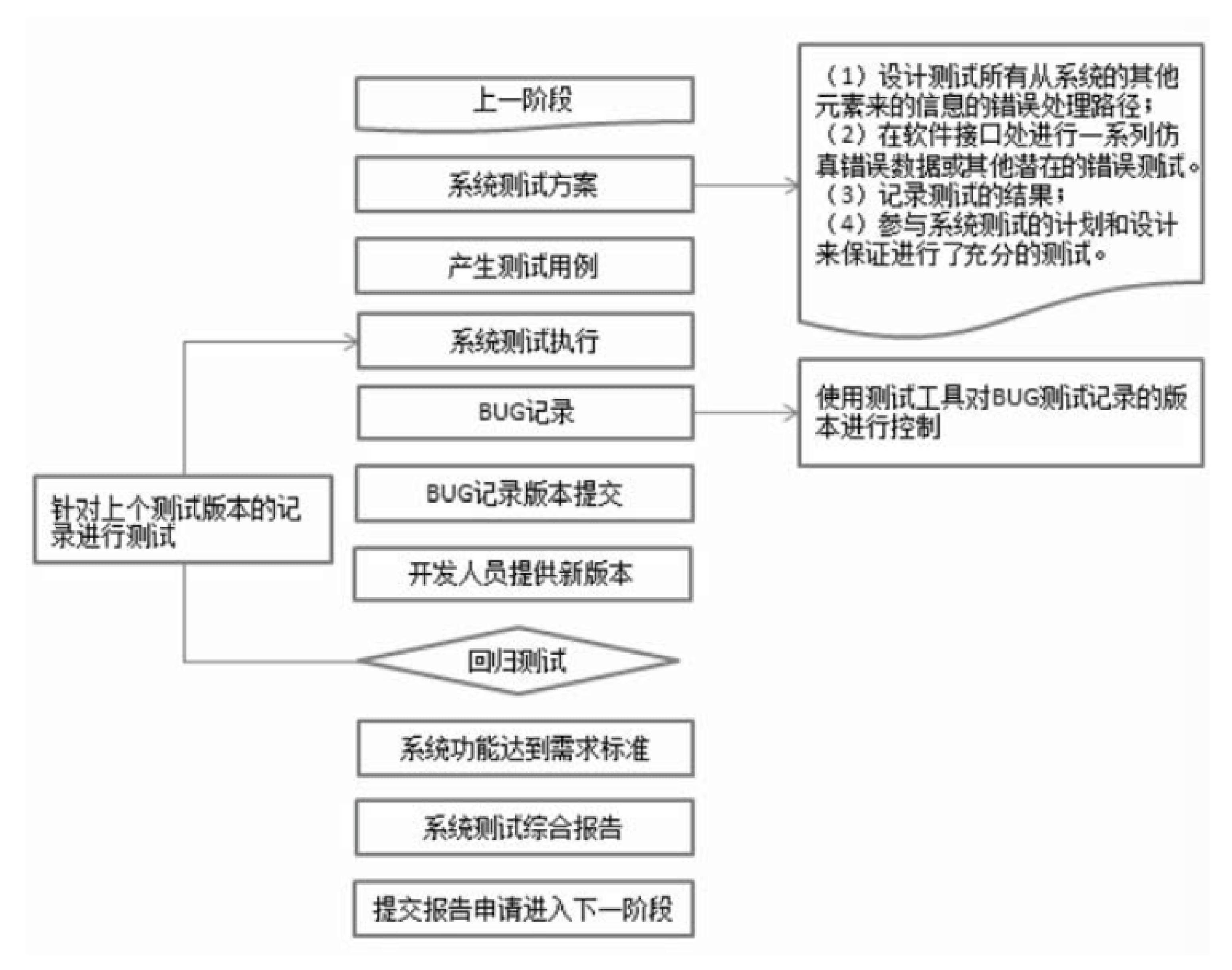


图 6-13　系统测试流程图

2. 性能测试

性能测试的流程有很多不同的环节。但主要的环节和顺序类似瀑布模型。其中包括：计划阶段、设计阶段、实施和执行阶段、报告结果阶段。每个阶段也很类似于其他测试类型的流程。图 6-14 是一个性能测试包括的主要环节和关系的实例。具体按阶段分析如下。

1) 性能测试计划阶段

测试计划阶段主要工作包括：明确测试对象、定义测试目标、定义测试通过的标准、规划测试进度、规划测试参与人员(需求、开发、测试、运维和配置)、申请测试资源和风险控制。

2) 性能测试设计阶段

测试设计阶段主要工作是测试用例设计、测试方法设计、选择测试工具、定义监控指标，如测试性能指标以及性能计数器等。

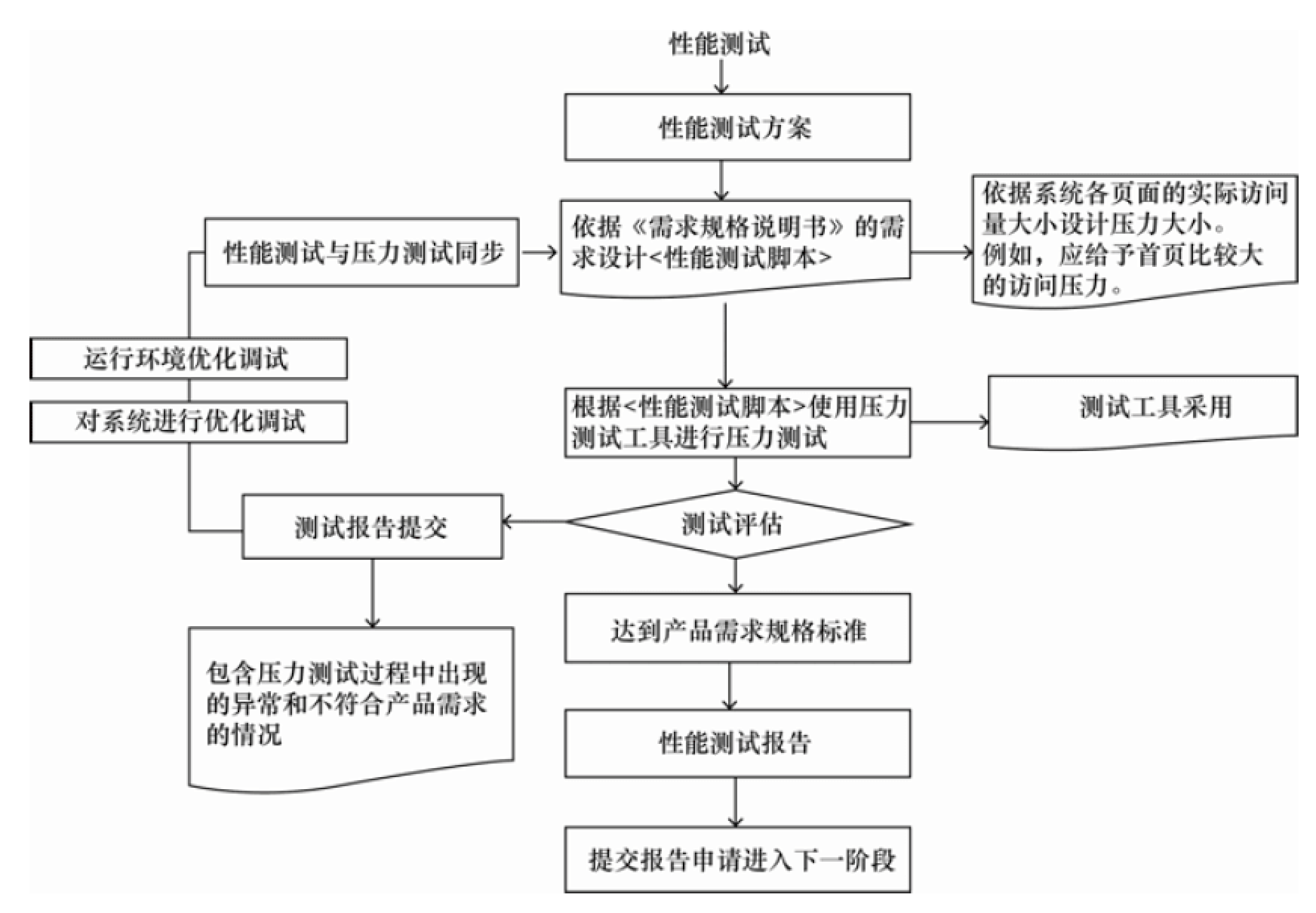


图 6-14　性能测试流程图

3) 性能测试实施阶段

测试实施阶段工作包括测试环境搭建、测试过程文档定义以及配置、测试脚本开发、调试、测试数据准备和基准测试。

4) 性能测试执行阶段

测试执行阶段工作主要是执行测试用例模型，包括执行脚本和场景。测试过程监控，包括测试结果、记录性能指标和性能计数器的值。

5) 性能测试结果分析和报告阶段

该阶段需要测试人员根据测试结果、记录性能指标和性能计数器的值进行测试分析。根据性能测试目标规划，分析出系统存在的性能瓶颈，并给出优化建议。还需要测试数据记录、性能指标以及性能计数器的值，给出对比以及总结性评价。

3. 自动化测试实施流程

自动化测试实施流程是描述软件功能自动化测试过程中的步骤、内容与方法，明确各阶段的职责、活动与产出物。图 6-15 自动化测试流程图几个主要环节及产出物说明如下：

与以前的测试计划过程一致，只是在原来的测试计划中，添加对项目实施自动化测试所需的资源、测试范围、测试进度的描述。该过程产出物为《测试计划》。

根据《测试计划》、《软件需求规格说明书》、《系统测试用例》设计出针对自动化测试的测试用例。测试用例的粒度精确到单个功能点或流程，对于各个功能点的业务规则，通过对脚本添加相应的检查点来进行测试。该过程的产出物是《自动化测试用例》。

根据《软件需求规格说明书》、《自动化测试用例》、《系统原型》、《系统设计说明

书》编写《自动化脚本设计说明书》，其主要内容包括：分析当前项目，设计出适合的脚本基本架构，针对特殊自动化测试用例设计可行的脚本编写方法，设计特殊检查点的实现方式，并对潜在的技术难点提出解决方案。该过程的产出物是《自动化脚本设计说明书》。

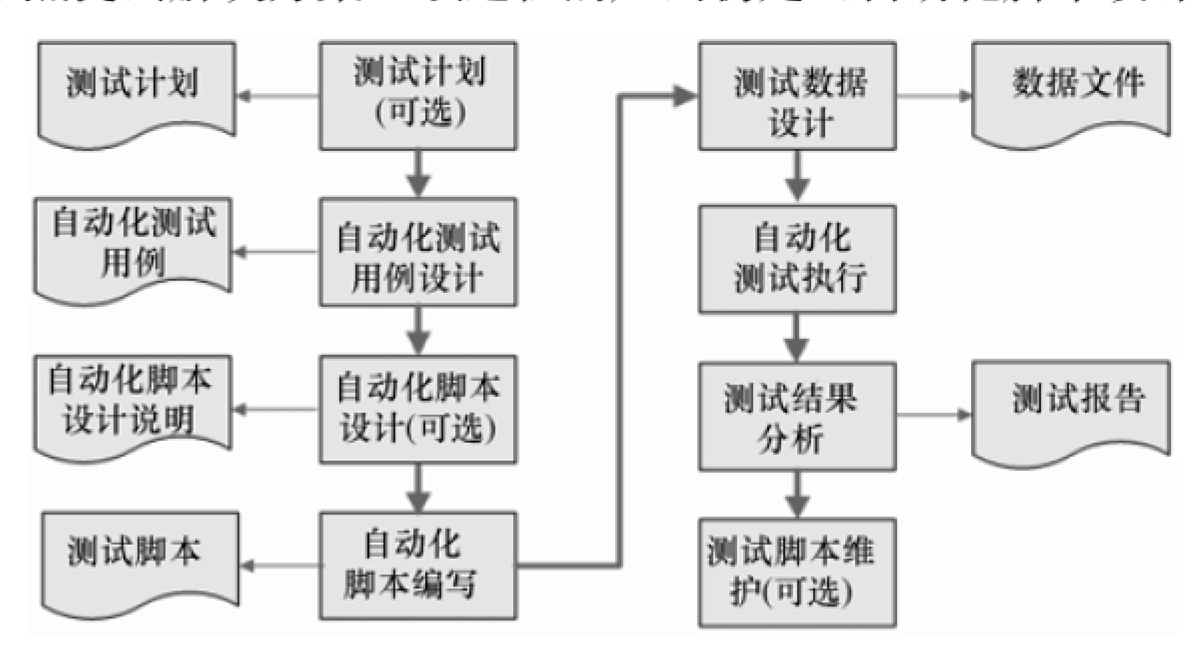


图 6-15　自动化测试流程图

根据《软件需求规格说明书》、《自动化测试用例》、《系统原型》、《自动化脚本设计说明书》，录制、调试、编写各个功能点的自动化测试脚本，并添加检查点，进行参数化。该过程还需要编写数据文件处理脚本、日志文件处理脚本、数据库处理脚本、公共检查点处理脚本等等。该过程的产出物是各个功能点的自动化测试脚本和其他公共处理脚本。

根据《软件需求规格说明书》、《自动化测试用例》设计出对各个功能点和相关业务规则进行测试的输入数据和预期输出，填写对应的数据文件。该过程的产出物是各个功能点的数据文件。

搭建好测试环境。根据《自动化测试用例》，执行自动化脚本，对系统进行自动化测试，并将测试结果自动记录到日志文件中。

对测试结果文件中报告错误的记录进行分析，如果确实是由于被测系统的缺陷导致，则提交缺陷报告。对自动化测试的结果进行总结，分析系统存在的问题，提交《测试报告》。

如果系统发生变更时，对自动化测试脚本和相关文档(包括《自动化测试用例》、《自动化脚本设计说明书》)进行维护，以适应变更后的系统。

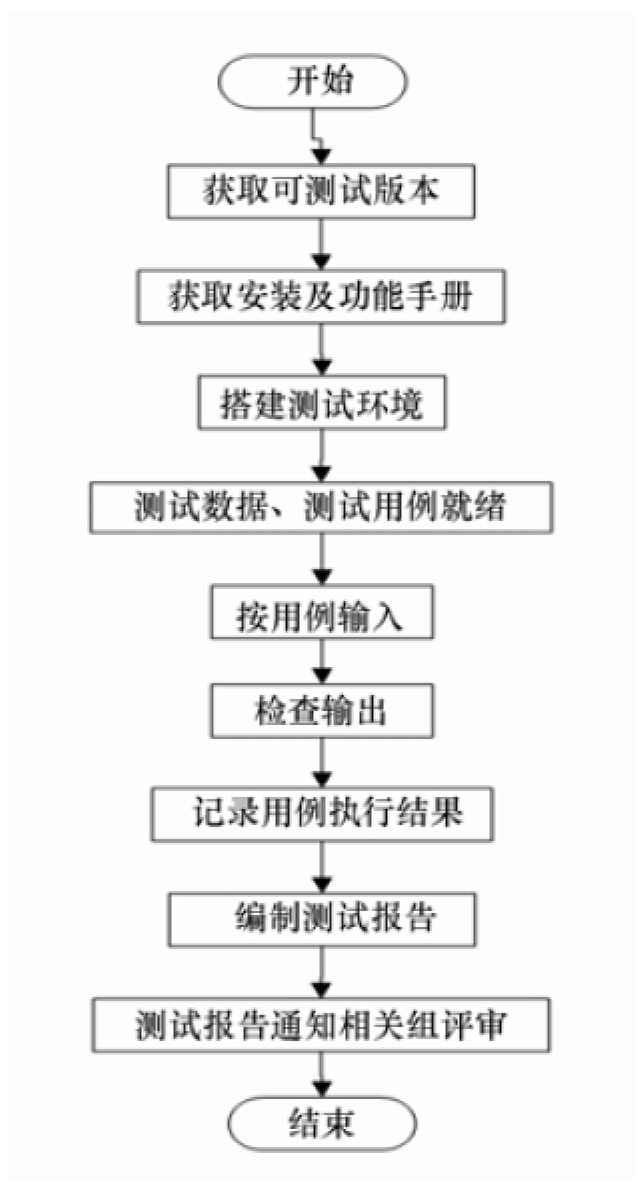


图 6-16　缺陷管理的流程图实例

4. 测试的执行

测试的执行也有流程图可以体现其考虑到的主要环节和内容。比如图 6-16 就是一个测试执行流程图实例。特别要说明的是测试执行活动最后一个环节，即结束或终止是重要环节。需要注意的是：

1) 正常终止：所有测试过程(或脚本)按预期方式执行至结束。如果测试正常结束，则核实测试结果。

2) 异常或提前结束：比如测试过程(或自动化脚本运行)没有按预期方式执行或没有完全执行。当测试异常终止时，测

试结果可能不可靠。在执行任何其他测试活动前，应确定并解决异常/提前终止的原因然后重新执行测试。如果测试异常终止，恢复暂停的测试。

5. 缺陷管理流程

这里从流程管理的角度，总体分析缺陷的生命周期。图 6-17 就是一个缺陷管理的流程图。体现很多相关要素的处理顺序和思路。

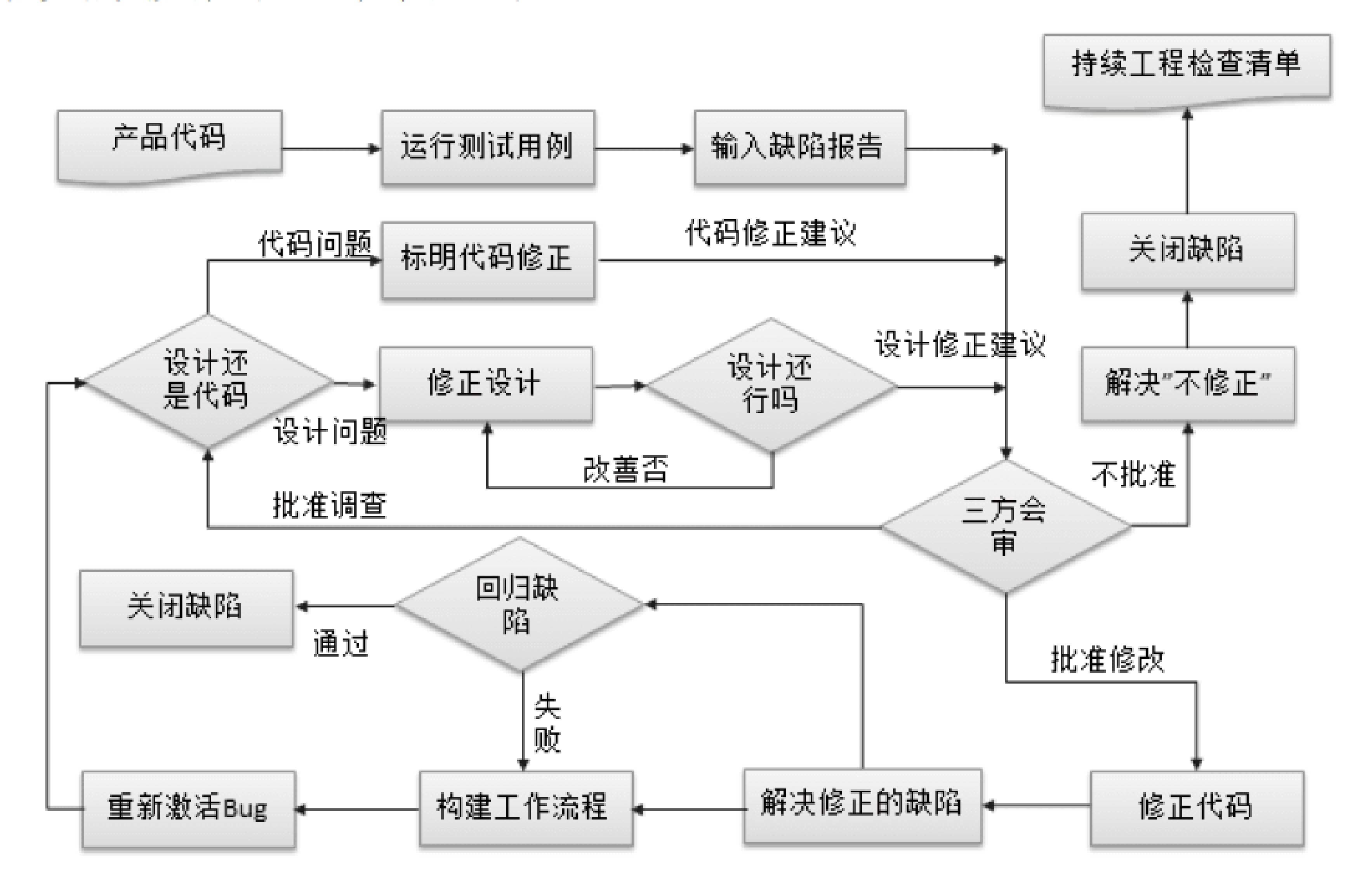


图 6-17　测试实施阶段流程

6.3　敏捷测试流程

由于近年来敏捷开发的实践被越来越多的企业采纳，敏捷测试的流程也成为很多业界专家研究的重点。所以专门占用一节的篇幅来分析。敏捷测试当然不能简单地理解为测得更快，绝对不是比以前用更少时间进行测试，也不是将测试范围缩小或将质量降低来减少测试任务。也有人说，只有敏捷开发，没有敏捷测试。假如将过去传统的测试流程和方法硬塞入敏捷开发流程中，测试工作可能会事倍功半，测试人员可能会天天加班，而不能发挥应有的作用。敏捷测试应该是适应敏捷方法而采用的新的测试流程、方法和实践，对传统的测试流程有所剪裁，有不同的侧重，例如减少测试计划、测试用例设计等工作的比重，增加与产品设计人员、开发人员的交流和协作。

Scrum 是一个用于运行项目的框架，它基于敏捷原则和价值。它定义一组活动，这些活动可帮助你的团队更快地向客户交付更多价值。 利用这些活动，客户有机会在你的团队开展工作时检查、指导和影响团队的工作。 此方法不会尝试在项目开始时定义所有内容。相反，你的团队以短小迭代(也称为“冲刺 (Sprint)”)为单位进行工作，并随团队工作的进展不断改

进计划。

整个敏捷测试过程中夹杂了很多在敏捷开发前已经出现的软件开发方法，包括极限编程(Extreme Programming, 1996)、Scrum(1986)、特征驱动开发(Feature Driven Development)，测试驱动开发(Test Driven Development)等。这些方法在敏捷软件开发流程的各个阶段都有充分的体现和应用。

例如，Scrum 主要着重于项目管理，团队中的项目经理(Scrum Master)需要在每个客户需求到来的时候制定 Sprint 的周期，定义每个 Sprint 的目标、分派任务、进行监督、最后总结得失并开始计划新的 Sprint。

相反，特征驱动开发和测试驱动开发主要应用于 Sprint 周期中。如果项目进行于开发新功能时期，这个阶段主要推行特征驱动开发。所有测试和开发人员都将自己的工作重心放在新功能上，从开发和测试两个方面来完成各自的任务。如果项目进行于测试新功能时期，这个阶段需要将工作重点挪到测试上来。所有测试和开发人员都密切关注着目前版本的缺陷状况。测试人员需要在每天的站立会议(Daily Standup Meeting)上报告前一个工作日发现的新缺陷情况，项目经理根据项目进度和缺陷严重性来决定是否修复这些问题。需要及时修复的缺陷是目前 Sprint 中的一个新任务，将由项目经理添加到 Sprint Backlog 上并通知开发人员去修复漏洞。

对于敏捷开发和测试中的审查过程，极限编程中的同行评审(Peer Review)思想得到了充分应用。代码和文档的审查追求简单而高效。团队成员两两组成一对，互相评审；有时，一个开发和一个测试人员也可以组成一对，互相协作。这样有助于缺陷和问题在第一时间被消灭在萌芽中。

6.3.1　敏捷测试流程的特点

敏捷开发模型主要适用场合是需求可能快速变化、开发周期短、发布频率快的软件项目。这类项目往往不等用户需求完全确定就已经开始着手进入研发，发布周期通常不超过四周甚至更短，然后再根据用户反馈调整用户需求，开发下一个版本。

敏捷测试也有着与传统测试不同的特点。它们体现在：

(1) 全程参与。敏捷测试更注重持续的对软件产品质量进行反馈。由于敏捷方法中迭代周期短，测试工程师应尽早开始测试。从项目开始讨论需求开始，测试工程师就应参与讨论，研究用户心理和用户行为，站在用户的角度对需求提出建设性意见。在设计阶段，对产品设计进行评审，同时开始准备用户案例(User Case)，作为后续阶段的铺垫。在实现功能阶段随时对新功能进行测试并执行回归测试。

(2) 轻量级文档。传统测试流程中，首先要建立项目的主测试计划书，然后针对每个功能领域制定测试计划，并与项目经理、开发团队讨论，集体认可签字才能通过，仅建立测试计划就要经过起草、评审和确认，往往要消耗数周的时间。在敏捷测试中，测试人员全程参与项目的研发过程，在每个迭代周期，只写出一页纸的测试计划，用于描述测试策略、测试范围和主要使用的测试方法学。

(3) 轻量级测试用例。传统测试流程中，测试工程师设计好测试用例后，要在测试用例管

理系统中详细记录，并严格按照测试用例描述的步骤执行。在敏捷测试中，测试用例的概念被淡化，直接根据用户用例(User Case)或者用户故事(User Story)辅以探索性测试开展测试。即使使用测试用例的形式，也采用粒度较大的测试用例，尽量用少量的测试用例覆盖更多功能。

敏捷测试的一个核心是迭代，在每个时间点上，所有项目人员都是有事可做的。敏捷测试的流程图如图 6-18。

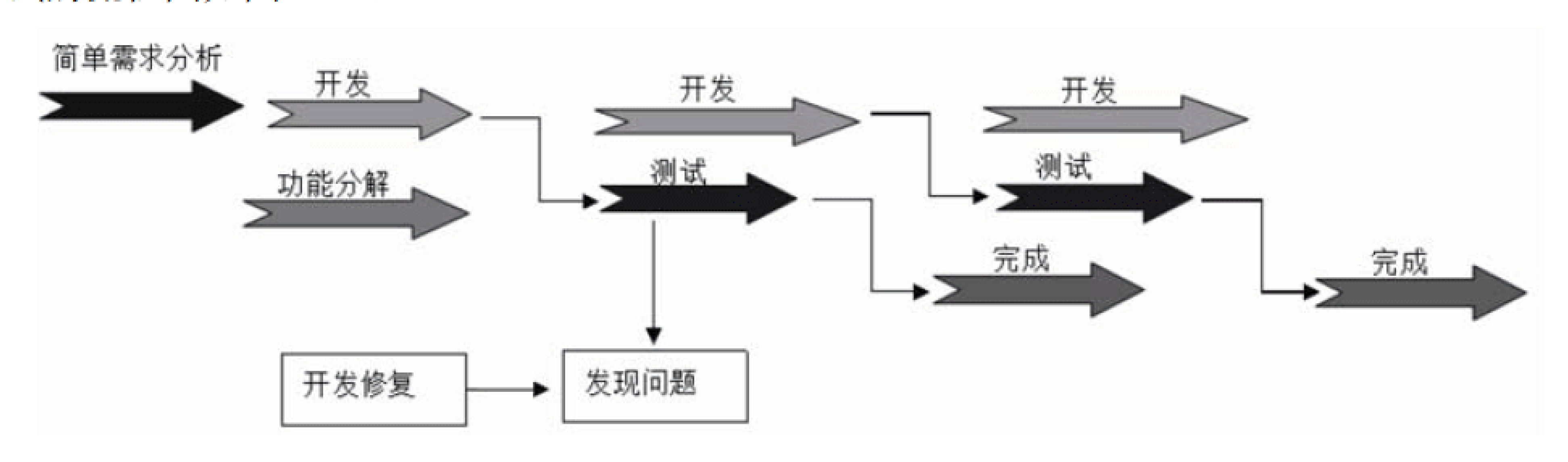


图 6-18　敏捷测试的流程图

具体说明如下：

第一阶段：通过上面的流程图，对于原本需要历时一个月的需求分析，在敏捷模式中，可能三五天就确定下来。这个需求制定得很模糊，但整体框架确定。产品对其中某一模块功能确认，开发人员开始对确认的功能编码，开发人员编码的过程中，测试进行功能分解，因为根据模糊的需求很难写出具体的用例，所以，只能尽量对功能进行分析得细些并标注需要验证的内容。

第二阶段：开发完成后交给测试人员进行测试，开发人员继续开发新功能。那么测试人员发现的问题怎么办呢？会从开发团队中抽调出一个人来解决测试发现的问题。但开发进度并没有因为测试而停止(如图 6-19)。

流程分析：在这个流程中弱化了文档，强调了各个人员的沟通，通过这种迭代的方式，三个月的项目，可能两个月和两个半月就会完成。但这种流程并不完美，假如一个功能在需求分析阶段就是错误的，因为它是一个迭代渐进的过程，也只能一路错下去。

其中对测试问题的处理也用一个流程图来清楚说明(如图 6-19)。

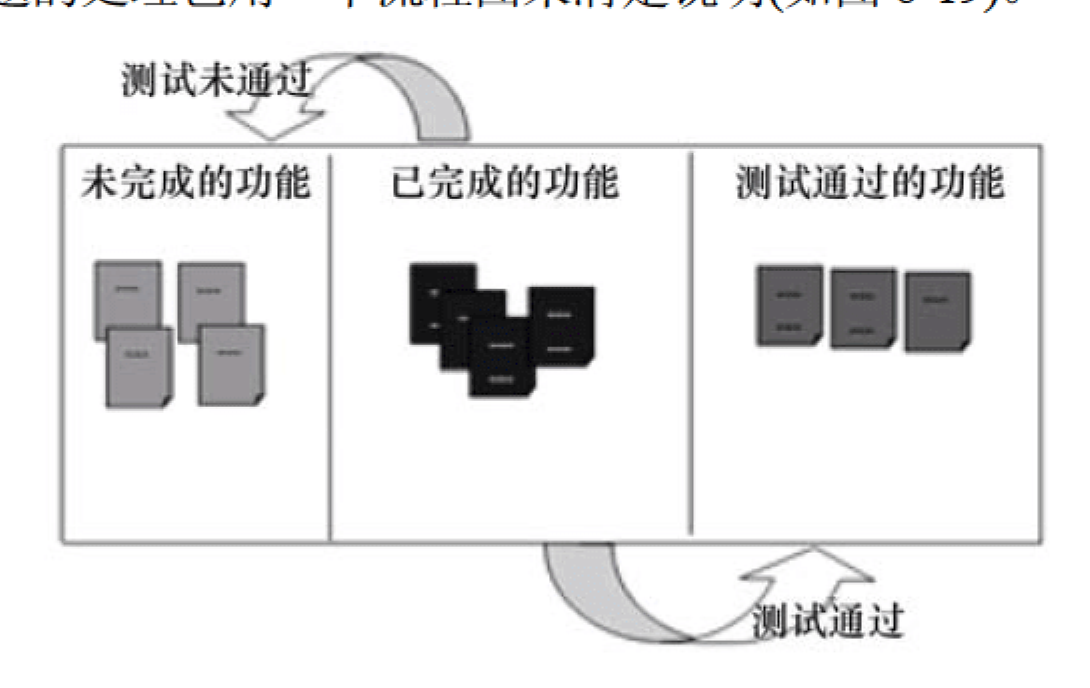


图 6-19　测试问题处理流程

第一块面板中是开发人员未实现的功能，第二块面板中是开发完成功能，测试人员对其

进行测试，发现不通过的就放回未开发的面板中，测试通过的将放到第三块面板中。

敏捷开发是一种以人为核心、迭代、循序渐进的开发方法。在敏捷开发中，软件项目的构建被切分成多个子项目，各个子项目的成果都经过测试，具备集成和可运行的特征。换言之，就是把一个大项目分为多个相互联系但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。敏捷测试是对应敏捷开发而产生的测试流程、方法和实践。与传统的测试流程相比，更重视测试工程师与产品设计人员、开发工程师的交流和协作，强化、突出测试工程师在研发过程中的作用，鼓励最终用户参与测试工作，更多的端到端(End-to-End)测试，测试计划和测试用例设计等工作的比重相应有所减少。

6.3.2　敏捷测试中的新功能测试和回归测试

一个软件项目研发过程中的测试工作可以分为针对新开发功能的测试和针对现存功能的回归测试。在敏捷测试中，针对这两种测试工作的策略有所不同：

1. 针对新功能测试的策略

以用户用例或者用户故事替代测试用例。持续进行验证，一旦一个具有完整功能的代码模块完成，立刻开始测试工作，而不是等待整个功能完全完成才着手测试。多实施端到端的测试，重视从最终用户角度出发保证业务流程的正确性和健壮性。在技术层面积极参与新功能的代码评审工作。

2. 回归测试的策略

敏捷开发模型中，迭代频率非常高，如果还按传统测试方法那样每次代码签入都要做一次完整的回归测试，无论人力还是其他资源都不足以支撑如此高的频度。所以一方面要实现更多的自动测试来保证回归测试的效率，另一方面对回归测试做适当裁剪。对回归测试的裁剪通常采用两种策略：一是通过代码变更区域的分析，明确知道某次代码签入影响的范围，只要对受影响的范围进行测试即可；另一种策略是用户关注程度和基于风险分析，对功能点进行优先级排序，必要时只测试高优先级的功能点，而忽视较低优先级的功能点。

完成对测试任务的估算，接着就可以开始详细设计验收测试用例。可对概要设计中的测试用例进行细化，根据不同的测试环境、测试数据以及测试结果，编写更详细的测试用例。另外，可以结合几个用例，完成一个复杂的测试操作。

由于敏捷开发的流程是不断迭代的过程，所以很多复杂的功能可能会在未来的 Sprint 周期中被优化。对测试人员而言，一个有效方法是尽量将一些验证基本功能的测试用例作为基本验证测试用例(Basic Verification Test Case)在第一时间实现自动化；而对一些复杂的功能测试用例，可先采用手工方法测试，直到在未来 Sprint 周期中该功能达到稳定时再考虑自动化。此外，对测试中出现的缺陷可以设计回归测试用例(Regression Test Case)，为其编写自动测试代码，使得此类问题在发布周期(Release Sprint)时可以顺利而高效得进行验证。

下面看一个项目实例：

项目介绍：根据一家在线 B2B 公司的要求，我们将为其开发一款类似于谷歌的搜索服务。作为 Web Service，该服务可以内嵌于网页中。当用户输入关键词并选择商户的类型和位

置后，系统会返回具体商户的列表(参见图 6-20)。



图 6-20　B2B 公司项目 UI 示意图

在用户故事和发布计划阶段，项目经理和产品经理会根据客户的需求，制定概要的产品发布日程计划。此时，测试人员可与开发人员一起学习新功能，了解客户的需求。其中有两个主要活动：寻找隐藏的假设和设计概要的验收测试用例。这种思考让测试人员更了解系统的隐含假设：

首先，系统应该能够在高峰时候处理 200 条搜索请求和 1000 个鼠标单击事件；

其次，用户可以在已经查找到的内容中继续查找；

最后，系统提供一个商户类别清单；如果用户选择商户类别而忘记具体名字，系统提供模糊查询。

在敏捷开发中，这些假设可作为用户故事记录下来，从而指导未来系统的开发和测试。

定义完一系列用户故事后，测试人员就可以着手设计概要的验收测试用例。正如我们在前文论述的，不同于单元测试，验收测试检查系统是否满足客户的预期，也就是用户故事是否能够实现。于是，测试人员可根据每条用户故事来扩展，寻找其中的“动作”，然后为每条“动作”制定正例和反例。

6.3.3　敏捷测试活动

典型的敏捷开发和测试活动参见表 6-1。它主要由三部分构成，从最初的用户故事设计和发布计划，到几次 Sprint 周期的迭代开发和测试，以及最后的产品发布阶段。每个时间段都有相应的测试活动。通常 Sprint 周期被分成两类：特征周期(Feature Sprint)和发布周期(Release Sprint)。特征周期主要涉及新功能的开发和各类测试。发布周期则会结合计划，确定新版本功能，然后对最新的功能进行测试。

表 6-1　敏捷开发测试活动表

敏捷开发的主要活动	测 试 活 动
用户故事设计	寻找隐藏的假设
用户故事设计	设计概要的验收测试用例
发布计划	估算验收测试时间
迭代 Sprint	估算测试框架的搭建
编码和单元测试	详细设计验收测试用例
重构	编写验收测试用例
集成	重构验收测试
执行验收测试	执行验收测试

(续表)

敏捷开发的主要活动	测 试 活 动
Sprint 结束	执行回归测试
下一个 Sprint 开始	发布

验收测试可以分为两大类，基本验证测试和功能测试。如果是基本验证测试，推荐开发人员在运行完单元测试和提交代码前直接运行自动测试脚本。如果是功能测试，可以在每个 Sprint 后期，新功能代码提交后，由测试人员单独执行。

在迭代的 Sprint 周期中，开发部分可根据传统步骤分成编码和单元测试、重构和集成。需要指出的是，重构和集成是敏捷开发的 Sprint 迭代中不可忽视的任务。如果在新的 Sprint 周期中要对上次的功能加以优化和改进，必然离不开重构和集成。

当一个 Sprint 周期正式开始时，项目经理将制定该周期的具体开发和测试任务。在定期的 Sprint 计划会议(Planning Meeting)上，每位团队成员都要提供自己在未来一个 Sprint 周期中的休假和培训计划。另外，每个团队可根据各自团队成员的能力和工作经验，适当设定一个工作负载值(Load Factor)。比如，我们团队的工作负载值为 75%，也就是说每个人平均每天工作 6 小时(以每天 8 小时计算)。接着大家就可以开始分配任务。

当开发团队开始编码和单元测试时，测试人员的工作重点包括：估算验收测试的时间、估算测试框架的搭建、详细设计验收测试和编写验收测试代码。第二个主要活动一般在项目初期的 Sprint 周期中完成。其他三个主要活动将在接下来的多个 Sprint 周期中视情况迭代进行。下面将具体介绍每个主要活动。

1. 估算验收测试时间

在软件开发初期，需要估算时间以便制定计划。这一点在敏捷开发中应用更广泛。如果以前的开发模式需要测试人员估算一个软件版本发行的计划(这样的计划通常会延续几个月)，那么现在则要在每个 Sprint 会议上估算两周到一个月的任务。此外，在每天的站立会议上，测试人员需要不断更新自己的估算时间，以应对变化的需求。所以，每个测试人员都应该具备一定的估算任务能力。下面将介绍两个通用的估算测试计划的方法。

1) 快速而粗糙的方法

从经验而言，测试通常占项目开发的三分之一时间。如果一个项目开发估计要 30 天 1 人，那么测试时间为 10 天 1 人。

以项目实例来说明：

搜索框的开发估计需要 78 天 1 人完成。但是，考虑到系统有模糊搜索的功能，所以测试任务可能会占 40%左右，大概 31 天 1 人。表 6-2 列出了具体任务。

表 6-2　测试任务与估算时间

任　　务	估计时间(天)
设计测试用例，准备测试数据(搜索数据集)	8
加载数据集	2

(续表)

任　　务	估计时间(天)
编写自动测试代码	18
执行测试和汇报结果	3
总结	31

2) 细致而周全的方法

这个方法从测试任务的基本步骤出发，进行详细分类。其中包括：

(1) 测试的准备(设计测试用例、准备测试数据、编写自动测试代码并完善代码)；

(2) 测试的运行(建立环境、执行测试、分析和汇报结果)；

(3) 特殊的考虑。

估算单个测试任务的事例参见表 6-3。

表 6-3　单个测试任务与估算时间

测试	准　　备		运　　行		特 殊 考 虑	估　　算
1	设计测试用例	0.5	建立环境	0.1		
	准备测试数据	0.5	执行测试	0.1		
	编写自动测试代码	0.5	分析结果	0.1		
	完善自动测试代码	2.5	汇报结果	0.1		
总计		4		0.4	0	4.4

估算多个测试任务的汇总参见表 6-4。

表 6-4　多个测试任务与估算时间

测试任务编号	准　　备	运　　行	特 殊 考 虑	估　　算
1	4	0.4	0	4.4
2	4	0.4	0	4.4
3	12	4.5	8.5	25
4	4	0.4	0	4.4
5	4	0.4	0	4.4
6	4	0.4	0	4.4
7	4	0.4	0	4.4
总计				51.4

2. 估算测试框架的搭建

测试框架是自动化测试必不可少的一部分工作。由于敏捷开发流程倡导快捷高效完成任务，这就要求一定的自动化测试率。一个完善的测试框架可以大大提高测试效率，及时反馈产品的质量。

在敏捷开发流程中，在第一个 Sprint 周期里，需要增加一项建立测试框架的任务。在随后的迭代过程中，只有当测试框架需要大幅度调整时，测试团队才需要考虑将其单独作为任务，否则可以不用作为主要任务罗列出来。

为此建立一个测试框架。在原先的估算中多增加一些任务。如表 6-5 所示。

表 6-5　敏捷测试任务与估算

测 试 任 务	估算(小时)
选择测试工具	3
建立测试系统	3
编写下载、存放和恢复测试数据的脚本	2
寻找或建立测试结果汇报工具	8
设计具体的搜索测试用例	4
准备搜索测试数据	4
编写和测试“搜索”模块	3
编写和测试“验证返回列表”的模块	1
学习“在结果中搜索”的模块设计	4
编写和测试“在结果中搜索”模块	4
第一次执行测试	4
分析第一轮测试结果	4
第二次执行测试	4
分析第二轮测试结果	4
总计	52

3. 详细设计验收测试用例

完成对测试任务的估算，接着就可以着手详细设计验收测试用例。可对概要设计中的测试用例进行细化，根据不同的测试环境、测试数据以及测试结果，编写更详细的测试用例。另外，可以结合几个用例，完成一个复杂的测试操作。

由于敏捷开发的流程是不断迭代的过程，所以很多复杂的功能可能会在未来的 Sprint 周期中被优化。对测试人员而言，一个有效方法是尽量将一些验证基本功能的测试用例作为基本验证测试用例，在第一时间实现自动化；而对一些复杂的功能测试用例，可先采用手工方法测试，直到在未来 Sprint 周期中该功能达到稳定时候再考虑自动化。此外，对测试中出现的缺陷可以设计回归测试用例，为其编写自动测试代码，使得此类问题在发布周期时可以顺利而高效得进行验证。

表 6-6 列出了基本验证测试用例。

表 6-6　基本验证测试用例

动　作	数　据	期待的结果
登录	用户名：(空) 密码：(空)	“用户名和密码无效”

表 6-7 列出了功能测试用例。

表 6-7　功能测试用例表

动　作	数　据	期待的结果
登录	正确的用户名和密码	进入系统：请输入搜索条件并单击“搜索”按钮
搜索	错误的类型	提示正确的类型
搜索	使用正确的类型	商户列表

敏捷开发不提倡撰写太多的文档，提倡直接编写测试用例。此外，测试人员和客户或项目经理取得良好的沟通，将这些需求总结下来，转化成验收测试用例。如果资源充足，最好对验收测试用例建立版本控制机制。

在每个 Sprint 周期结束前，测试团队将提交针对该 Sprint 周期或者上个 Sprint 周期中已完成的功能的验收测试(在实际项目中，测试团队的进度通常会晚于开发团队)。这样，开发团队可以运行验收测试来验证所开发的功能目前是否符合预期。当然，这个预期也是在迭代中不断变化和完善的。

在一个 Sprint 周期结束时，团队要举行一个回顾会议(Retrospective Meeting)。团队成员可以在会议上畅所欲言，指出在过去一个 Sprint 周期中可行的、不可行的和有待改进的地方。待改进之处将在项目经理监督下于未来的 Sprint 周期中实现。

由于敏捷开发倡导增量开发，当新的 Sprint 开始时，测试团队需要根据新 Sprint 周期的开发进度及时重构验收测试。如果新 Sprint 周期没有具体的新功能开发，测试团队可将精力集中在执行验收测试和寻找缺陷上。

6.3.4　敏捷测试中的测试工程师

本部分将简要介绍敏捷开发中测试人员所需要具备的素质和职责，还介绍了工作流程。

1. 敏捷开发团队介绍

常见的敏捷开发团队由四位开发人员、两位测试人员、一位产品设计、一位项目经理和一位产品经理组成。每天早上在固定的时间和会议室里团队举行站立会议。这时候，团队成员按照既定的顺序向项目经理汇报各自前一天完成的任务、所遇到的困难和当天要完成的任务。同时，项目经理更新 Sprint Backlog(一张制作精良的 Excel 表格)，并及时解决每个人所提出的问题。由于敏捷开发要求参与人能够快速而高效地应对变化，所以无形中对测试人员提出很高的要求。

2. 测试人员需要具备的素质

测试是软件开发中不可或缺的一部分。在敏捷软件开发中亦是如此。不同的组织给测试人员以不同的称号：测试开发(Test Developer)、质量分析员 (Quality Analyst)、软件质量工程师(Software Quality Engineer)等。每个称号隐含有不同的职能。以上的称号分别对应以下的能力要求：

1) 具有质量检测和编写代码的能力——>软件质量工程师

2) 具有防止缺陷和质量控制的能力——>质量分析员

3) 具有开发和执行测试程序的能力——>测试开发

总结起来有三方面的基本素质要求：代码编写(Coding)、测试(Testing)和分析(Analysis)。

在很多其他的开发流程中，各个测试阶段对测试人员的能力有所不同；有时侧重分析(比如系统配置测试)，有时侧重代码编写(比如功能测试)。但是，在敏捷开发流程中，测试人员需要结合这三方面来开展工作，只有这样才能真正反应敏捷测试的本质：简单而高效地应对变化。

3. 测试人员的主要职责

在敏捷软件开发中，测试人员的职责有三个主要方面：

1) 定义质量：这应该是软件测试人员的基本职责。敏捷方法鼓励测试人员在 Sprint 计划的时候直接与客户交流，从自己的经验出发，共同为产品功能制定质量要求。

2) 交流缺陷：敏捷过程强调团队中的交流。开发人员经常会专注于重要而新奇的功能，测试人员应该抓住细节，寻找设计中的 Missing Door；另外，开发人员使用单元测试来保证产品的基本质量，测试人员可以使用验收测试(Acceptance Test)来鉴定客户需求与实际成果之间的不一致性。

3) 及时反馈：敏捷过程强调简单而高效。测试人员需要及时反馈产品目前的质量问题。这样一来，团队才可以立刻着手解决。如果传统的流程是一周汇总一次状态的话，敏捷流程要求每天汇总质量问题。在我们的项目中，内部的测试报告会以网页的形式显示在内部站点上。每个团队成员能够随时获取。另外，我们的测试框架提供自助测试(Self-Assistant Test)：通过单击测试用例列表中的某个具体用例，开发人员不需要中断测试人员的工作就可以重现缺陷。

习题与思考题

1. 测试流程管理的意义？

2. ISTQB 定义的软件测试的一般流程？

3. 敏捷测试的流程是怎样的？有什么特点？

4. 敏捷测试中测试工程师的职责是什么？

5. 简述测试进度的影响因素和分析怎样规避风险，保证进度？

6. 开放式问题：在某网上商城的测试执行阶段，负责测试的测试工程师发现商品页面中多了一个“赞”的功能，这个功能在开发和测试文档中都没有描述，这名测试工程师应该怎么做？

第7章　软件测试执行管理

测试管理需要覆盖测试过程中所有的测试活动。软件测试执行的环节是所有测试活动中最基本也是软件测试任务中的最能体现最终结果和质量的一环。换句话说，测试主要侧重于测试的执行。实际工作中，当自动化测试工具在功能测试中难以发挥作用时，测试执行的工作量会很大。测试的执行是充满创意的活动，因为在设计和执行测试的过程中都需要测试人员应用自己的理论和技术知识，理解测试计划、范围、案例的预期结果和要求。测试执行管理是基于整个测试执行过程开展的。

学习完本章后，学员应具备以下能力：理解软件执行管理的要素，了解业界常用实践。领会不同软件执行活动需要注意的环节和要点，掌握应用惠普相关工具的动手能力，明确怎样根据测试计划决定怎样执行测试，并努力做到高效和高质量地完成测试任务。

7.1　软件测试执行基础

软件测试执行阶段(业界有些也称为测试的实施阶段)是要在前面软件测试计划的基础上有效地按照事先预定的测试计划、设计的测试用例、自动化测试运行、测试完成标准等具体操作的过程，并最后报告结果，相当于验收的具体过程。这就是软件测试执行管理的基本目标。软件测试的执行是非常重要和关键的环节。测试执行(Test Execution)是对被测组件/系统执行测试，并产生实际结果的过程。要做好软件测试执行的管理，必须了解测试执行的内容、主要环节、影响因素等基本知识和方法。

7.1.1　软件测试执行的内容

测试执行包括手动测试和自动化测试。常见的测试执行是测试环境搭建之后，运行已有的自动化测试脚本，手动测试自动化测试未覆盖的测试用例或任何需要测试的场景。

软件测试执行的内容就是要决定怎样执行测试和测试什么。根据测试项目的需求，测试范围和交付的标准等决定。并没有统一的答案。决定测试执行的内容需要明确以下信息：

1. 测试执行依据的文档

测试执行根据《测试需求文档》、《测试计划》、《测试执行计划》和《测试用例》等文档规定的内容执行。

2. 制定测试执行计划

测试执行可以预先制定具体怎样操作的计划，以保证正常的实施测试执行活动。

注意考虑的内容包括测试执行的具体安排。比如：

1) 测试执行的时间安排
2) 测试执行的人员分配
3) 测试执行的环境要求和搭建

3. 记录测试执行的结果

1) 做测试执行记录的理由

(1) 保证测试工作的可追溯性
(2) 记录测试人员的工作情况
(3) 测试人员绩效评估的重要数据

2) 记录的内容

(1) 谁负责——测试执行人员
(2) 什么时候——日期、时段
(3) 做了什么——测试用例、自动化脚本
(4) 结果如何——通过/失败

4. 执行测试的过程

1) 设置测试环境，确保所需的全部构件(硬件、软件、工具、数据等)都已实施并处于测试环境中；

2) 将测试环境初始化，以确保所有构件都处于正确的初始状态，可以开始测试；

3) 运行自动化测试和执行手动测试用例。

5. 测试执行活动结束或终止

1) 正常终止：所有测试过程(或脚本)按预期方式执行至结束。如果测试正常结束，则核实测试结果；

2) 异常或提前结束：测试过程(或脚本)没有按预期方式执行或没有完全执行。当测异常终止时，测试结果可能不可靠。在执行任何其他测试活动前，应确定并解决异常/提前终止的原因然后重新执行测试。如果测试异常终止，则恢复暂停的测试。

6. 核实测试结果并报告缺陷

1) 通常发生在实际结果与预期结果不匹配时；
2) 意外的 GUI 窗口(图形界面)；
3) 业务流程错误；
4) 报告其他任何异常和设计缺陷。

7. 测试执行的准备

1) 执行前，动员工作：阐述策略，回答问题，定义测试计划、测试范围；

2) 严格审查测试环境，包括硬件、网络拓扑结构、网络协议、防火墙或代理服务器的设

置、服务器的设置、应用系统的版本；

3) 将测试用例分类进行有效整合，构造测试套件(Test Suite)；

4) 所有测试用例、测试套件、测试任务和测试执行结果通过测试管理系统进行管理，使测试执行的操作过程记录在案，能跟踪、控制和追溯，保证测试进度和质量；

5) 确保每个测试人员理解测试策略、测试目标，对测试进程进行审查，确保测试策略得到执行，可采取奖励手段，测试经理、组长要勇于承担风险，使测试人员有发挥、想象的空间，但同时也施加压力，提高工作效率和责任心。

8. 测试执行过程

1) 记录测试记录，测试用例的执行状态注明 Pass/Fail；

2) 记录缺陷报告捕获测试结果，提交给上级审查及实现人员进行修复；

3) 缺陷跟踪和管理一般由特定工具来执行，对各模块、各测试人员、整体项目等进行事实跟踪；

4) 进行常规的缺陷审查，包括 Bug 的严重性、Bug 的描述、Bug 修正等；

5) 验证缺陷的修复，执行回归测试；

6) 对每个阶段测试结果进行分析，保证阶段性的测试任务得到完整的执行并达到预定的目标；

7) 良好的沟通，测试人员之间，与项目组之间保持有效的沟通，比如把每周例会作为测试执行的流程环节，可以及时发现测试中的问题，及时沟通并快速解决。

7.1.2　影响测试执行的因素

测试执行处于软件测试生命周期的关键路径上，它不仅在测试过程中占有重要地位，也会花费大量测试时间。针对测试执行而进行的计划，即测试执行进度计划，是进行测试执行进度控制的基础。在进行测试执行进度计划制订时，需要考虑哪些因素会影响测试执行活动，以及如何针对不同类型的测试人员进行测试执行进度计划的调整。

在实际的软件测试过程中，测试资源、测试质量、测试时间之间是相互制约的。测试执行进度计划的制订，需要在这三者之间进行平衡。例如：假如项目产品发布的时间是确定的，或者根据市场或客户的需求是受到限制的，那么在有限的时间内，需要在有限的测试人力资源和其他的测试资源与测试质量之间进行平衡，即测试执行进度计划的制订，需要我们在有限的测试时间内，利用现有的测试资源，达到既定的测试质量。

1. 软件测试执行影响因素

对执行软件测试影响因素最大的是以下几方面：

1) 测试计划(Test Plan)：测试计划是测试策划过程的一份记录。要完成好测试执行任务，必须要理解和明确预先制定的软件测试计划。因为软件测试计划包括重要的测试范围、测试目标、策略、工具、测试交付标准等重要信息。这些信息是指导软件测试具体怎样执行的参考因素。

2) 测试环境准备(Test Environment Preparation)：测试人员需要搭建和维护测试环境，保

证测试执行环境和测试管理环境(例如：配置管理、缺陷管理等环境)可用。测试环境应该在测试执行之前完成搭建和相关的验证工作。对于高级别的测试，例如：系统测试，测试环境应该尽量和用户的使用环境接近，能够模拟用户场景。

3) 测试实现(Test Implementation)：测试实现是开发、排序测试规程，创建测试数据，准备测试工具和编写自动化测试脚本的过程。

业界常用测试实现和测试执行两个基本概念描述测试的具体执行，但有些专家认为不需要具体划分这两个概念，而是把两者的内容都认为是软件测试执行的范畴。

4）人为因素：测试的执行都是由人来操作的，所以存在很多不确定的因素。总之，测试计划相当于测试执行要遵循的法。它规范了测试人员在执行测试过程中遇到的“测试什么”和“如何测试”的问题。测试计划的范围和内容直接影响到测试的执行及结果。因此测试计划的制定对于执行测试至关重要。

2. 测试执行进度计划的影响因素

在制订测试执行进度计划的时候，至少需要考虑下面的这些因素：过程的成熟度、测试的时间范围、测试的资源、产品的质量、测试的文档等。

1) 过程成熟度

影响测试执行进度计划制订的第一因素是组织或者说是团队的过程成熟度，包括开发过程的成熟度和测试过程的成熟度。软件产品的质量需要通过整个软件开发过程来保证，而不是某个人或者某部分人的职责。因此，组织的过程成熟度会直接影响测试执行进度计划的制订，具体表现在：

(1) 开发过程成熟度，直接决定了开发得到的工作产品的质量，比如软件的需求文档、设计文档、代码等质量。任何软件工作产品都可能是我们的测试对象，同时也是我们测试的基础。这些工作产品的质量，会直接影响我们的测试工作量和测试执行进度计划的制订；

(2) 测试过程成熟度，决定了主要的测试活动和测试阶段，简单地说，测试执行过程中我们应该做什么。对于测试执行阶段，不同的测试过程成熟度，需要采用的测试活动是不一样的。例如：有的测试执行明确定义了正式测试执行之前的预测试；而有的测试执行可能更强调回归测试。而这些因素，都会影响测试执行进度计划的制订。

测试过程的成熟度，也会影响测试执行过程中的输出工作产品，例如：缺陷报告、测试总结报告等。对这些文档的要求，也需要在测试执行进度计划中进行考虑。

过程成熟度定义的测试执行相关的度量，例如：测试用例执行的速率(测试用例数目/星期)、测试的有效性(缺陷数目/测试用例)等，它们是进行测试工作量估算的基础，因此在测试执行进度计划制订中需要谨慎考虑。

2) 测试的时间

测试时间是制订测试执行进度计划的基础。这里的测试时间，指的是测试执行需要在什么测试时间范围之内完成。在有的项目测试执行过程中，软件测试执行的截止时间是确定的，例如：软件产品必须在 2007-12-31 之前交付给客户，所有测试活动都必须在这个时间之前完成。这种情况下，测试的时间范围已经无法进行选择，我们必须在测试资源、测试质量和测

试范围等方面进行平衡。

3) 测试的规模

在制订测试执行进度计划的时候，需要详细考虑测试对象的规模。测试对象的规模是我们进行测试工作量估算的基础，这同样适合测试执行的测试任务。针对测试执行的测试规模，可从下面几个方面进行考虑：

(1) 测试执行过程中需要执行的新设计的测试用例的数目；

(2) 测试执行过程中需要执行的回归测试用例数目；

(3) 测试对象中可能存在的缺陷数目，以及针对这些缺陷可能需要进行的回归测试；

(4) 测试用例的执行是针对多种不同的测试平台，还是只针对一种测试平台。

4) 测试的资源

在确定了测试时间和测试规模后，接下来需要考虑测试执行涉及的测试资源问题。测试资源的范围很广，包括测试人力资源、测试仪表、测试平台等。

测试执行活动需要有合适的测试人员来完成。根据组织内已经定义的相关度量或者项目组成员以前的经验值和估算测试规模大小来确定需要的测试人员数目。

测试执行过程对测试平台数目和已有测试平台数目有要求。假如测试平台数目无法满足测试执行的要求，需要在测试执行进度计划中体现，因为很可能需要测试时间来解决这个问题。测试执行过程中需要使用的测试仪表和已有的测试仪表。和测试平台一样，也需要在测试执行进度计划中体现，并提供合适的解决方案。

5) 产品的质量

前面的“过程成熟度”部分已经涉及测试文档和测试对象的质量问题。这里将更详细地讨论它们是如何影响测试执行进度计划的制订：

(1) 开发文档的质量：是指在开发过程中输出的文档质量，比如软件需求文档、概要设计文档、详细设计文档等。由于这些开发文档是测试设计的基础，同时设计得到的测试相关文档是测试执行的基础，它们将直接影响测试执行的效率和有效性。

(2) 测试文档的质量：一方面开发文档的质量会影响测试文档的质量，另一方面，测试人员的技能水平和在项目相关领域的背景知识等都会影响测试文档的质量，从而影响测试执行的效率和有效性。

(3) 软件代码的质量：开发文档的质量和开发人员的技能、知识水平能力会影响软件代码的质量。而软件代码是我们测试执行运行的真正的对象，它的质量高低直接影响了测试执行进度计划的制订。比如由于测试对象质量很差，导致测试执行并不是原来计划中的一次，而需要进行多轮的测试。

测试执行对象中可能存在的缺陷数目以及后续的回归测试，它们在测试执行进度计划制订过程中经常容易被忽视。测试执行过程中发现的缺陷，在修改完成后，需要测试人员进行验证测试和相关的回归测试。

如果在测试执行进度计划中没有考虑这些质量因素，常常会导致测试后期的测试任务非常繁重，从而影响测试执行的效率和测试质量。

6) 测试的文档

测试执行过程和测试执行完成后，都需要输出一些测试相关的文档，比如测试过程中需要提交的缺陷报告、测试执行结束之后提交的测试总结报告和测试版本发布报告等，这些文档都需要测试人员花费时间和工作量来完成。

测试实现和执行阶段的主要活动包括：通过特定的顺序组织测试用例来完成测试规程和脚本的设计，并且包括测试执行必需的其他任何信息，以及测试环境的搭建和运行测试。

根据测试计划、测试设计规格说明、测试用例规格说明，结合各个测试用例之间可能存在的依赖关系，设置测试执行的顺序，例如：根据业务过程，设置执行顺序；根据测试用例的优先级设置执行顺序。

测试用例执行时必须选择合适的测试数据，在有些测试中，测试数据甚至是非常庞大的。因此，在测试实现阶段，测试人员需要将输入数据转换成相应的数据库。同时，测试人员也可能需要编写脚本生成测试数据，在测试执行的时候作为软件系统的输入。

假如采用自动化测试，测试实现还包括了自动化测试套件和测试脚本的创建和开发。测试人员在测试实现阶段，还应该考虑一些具体的可能会影响测试顺序的制约因素，以及测试环境和测试数据之间的相互依赖关系。

7.1.3　软件测试执行的管理

测试执行的管理应该针对测试执行的所有环节。从一开始决定测试执行的内容到怎样执行测试、再到明确测试执行的预期目标和交付结果等，都要遵照规定的流程和规范进行。也要处理好时间的管理。本节先介绍戴明环，再以测试执行的起始和结束为例说明测试执行管理要考虑和关注的环节。

1. 戴明环指导测试执行

PDCA 循环又叫戴明环，是美国质量管理专家戴明博士首先提出的，它是全面质量管理所应遵循的科学程序。全面质量管理活动的全部过程，是质量计划的制订和组织实现的过程，这个过程就是按照 PDCA 循环，不停顿地周而复始地运转的。

如图 7-1 所示的 PDCA 循环的四个阶段和 8 个步骤应该作为测试执行管理的指导思想。

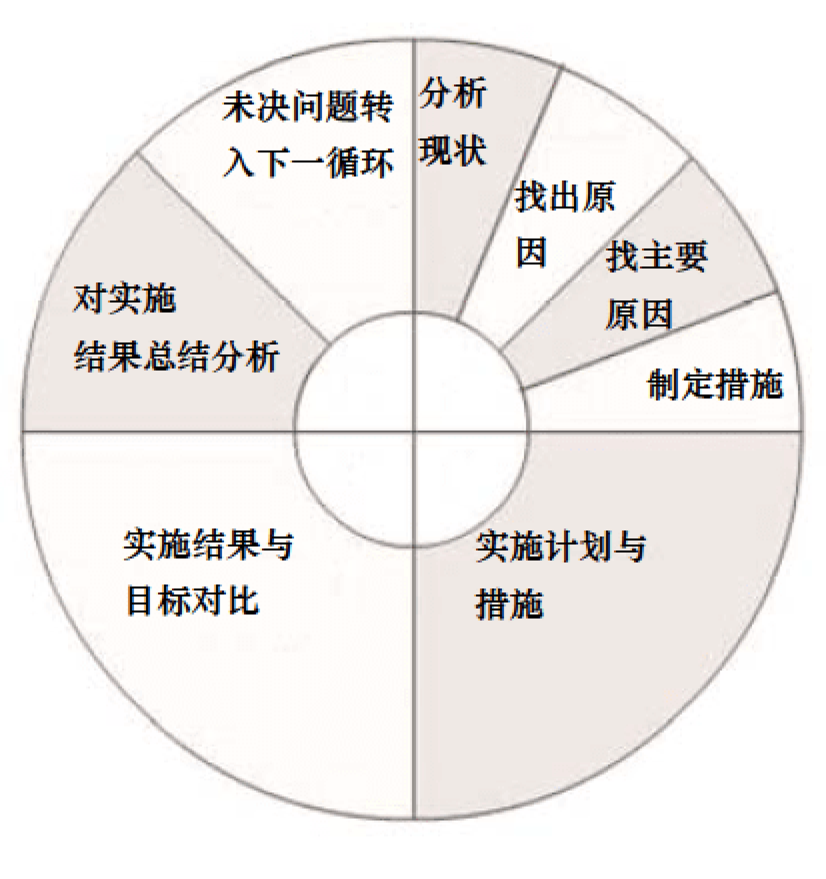


图 7-1　PDCA 原理

PDCA 是英语单词 Plan(计划)、Do(执行)、Check(检查)和 Action(处理)的第一个字母，PDCA 循环就是按照这样的顺序进行质量管理，并且循环不止地进行下去的科学程序。PDCA 循环的 8 个步骤是：找问题、找原因、找主要原因、制定计划措施、实施计划、检查、总结经验、提出新问题。应用于测试执行的管理就是要在测试执行前就认真分析要执行测试的范围，包括手动和自动化的测试用例、测试执行人员的分工合作、需要遵守的测试执行流程、测试执行结束要交付的

结果、其中发现 Bug 的处理方式等等。明确测试执行的目标，制定测试执行的策略和计划，以及阶段检查的时间表。在测试执行过程中积极思考问题、原因、解决办法，及时总结经验教训，调整测试执行的实施，努力提供测试执行水平，以保证测试任务的顺利和高质量完成。测试执行完成后总结经验教训，推广最佳实践。注重测试人员能力的提升。

2. 测试执行的起始

当测试对象满足测试执行的入口准则时，测试执行就可以开始了。测试执行应该按照确定的测试顺序进行。测试执行活动的一个核心内容是对测试结果和期望结果进行比较。假如测试结果和期望结果不符，首先需要仔细检查测试用例，确保设计的测试用例是正确的。测试用例描述也可能存在错误，原因是多方面的，例如：测试数据的错误、测试文档描述的错误或者测试执行方法错误等。假如测试用例的描述存在问题，则首先需要对它进行修改，然后重新执行测试用例。如果确认是测试对象的问题，就需要提交缺陷报告。

1) 记录测试执行结果

在测试执行阶段，测试执行过程和结果必须妥善进行记录。执行过的测试用例，如果没有记录测试结果，很有可能出现重复执行的情况，从而导致测试效率低下和测试进度延期。由于测试对象和测试环境会随着被测试版本变化而变化，所以测试记录应该基于相应的测试版本。

测试日志提供了按照时间顺序的测试执行相关细节。测试结果的记录可以针对整个测试过程，也可以针对某个事件。任何影响测试执行的事件都需要单独记录。为了测量测试覆盖率和查找测试延期的原因，需要测试人员记录详细的测试信息。另外，记录的信息也有助于测试控制、测试进度报告生成、测试出口准则评估和测试过程的改进。

测试执行过程中，建议测试对象的用户或客户也尽可能参与测试执行，例如：验收测试过程中，发现的被测试产品的缺陷很少，有助于客户对软件系统建立信心。

2) 测试执行的流程

如图 7-2 所示，测试实施和执行有很多环节，相互之间有些有依赖关系。

测试实现和执行阶段的主要输入有：

(1) 测试计划
(2) 测试需求
(3) 测试设计规格说明
(4) 测试用例

测试实现和执行的主要测试活动有：

(1) 创建测试数据
(2) 编写测试规程规格说明
(3) 开发测试自动化脚本
(4) 根据测试规程规格说明创建测试套件
(5) 搭建并验证测试环境
(6) 执行测试用例，包括手工和自动化执行
(7) 记录测试执行的过程和结果(测试日志)

(8) 测试实际结果和期望结果的比较

(9) 提交缺陷报告

(10) 确认测试和回归测试

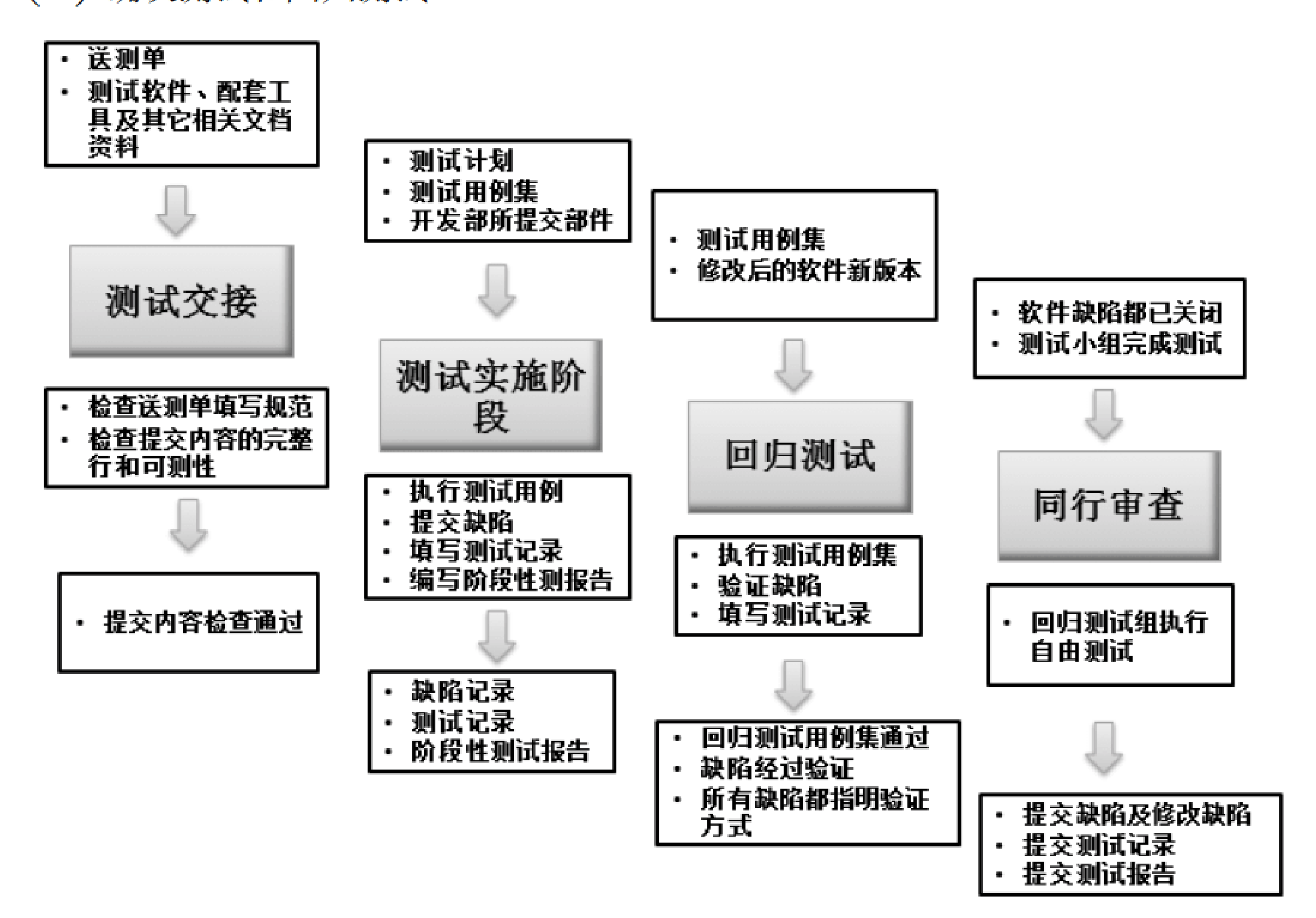


图 7-2　测试实施阶段主要环节和顺序图

评审测试计划内容的正确性及合理性：

(1) 测试环境、测试资源

(2) 测试需求范围，各个测试需求的优先级

(3) 测试策略及风险管理等

(4) 评审测试用例集

(5) 测试用例优先级

(6) 测试用例集基于需求的覆盖程度

在测试执行中，测试人员按照测试规程和测试用例中的步骤执行测试，将实际结果和期望结果进行比较。假如是由于测试对象的原因导致测试结果和期望结果不符，那么测试人员需要提交缺陷报告。测试执行过程中的测试状态和测试结果必须妥善进行记录，避免由于没有记录测试结果导致的重复执行测试用例的问题。测试执行过程中还需要收集相关的数据，例如：缺陷相关的数据、测试执行的数据等，通过分析和评估收集的数据对测试过程进行有效监控。

随着测试的进行和对程序的理解加深，测试人员在执行测试用例时会发现已有的部分测试用例集不充分或者不理想。测试团队需要酌情更新测试用例或及时把情况和建议汇报给测试负责人。测试负责人需要相应调整测试执行计划和进度安排。

在条件允许的情况下，可以让用户或客户参与到测试执行活动中来。用户或者客户参与

测试执行不仅有利于尽可早发现产品功能和用户期望之间的差距，也可以作为增强用户或客户对系统信心的一种手段。在测试执行过程中，产品运行良好，缺陷很少，会极大地提高用户或客户使用该产品的信心。通常用户或客户参与的都是比较高级别的测试，例如：系统测试和验收测试。这样更有利于用户或客户对系统提出意见和建议，以及获得对产品的信心。

3) 测试执行入口准则

测试执行入口准则指的是允许软件系统或者软件产品进入测试执行阶段所必须具备的条件。也就是说，提交的软件系统或者软件产品必须满足入口准则定义的条件，测试团队才可以进行测试执行的具体工作。入口准则的定义可以考虑各个方面的因素。

示例：系统测试执行入口准则

针对系统测试，可以定义下面的一些条目作为测试执行的入口准则：

(1) 测试设计规格说明和测试用例规格说明已经编写完成，并且通过评审。

(2) 自动化测试用例验证通过。

(3) 相关的测试资源和测试环境准备就绪，包括人员、工具、实验室等。

(4) 开发人员对提交的版本进行了预测试，并且预测试通过(也可能是测试团队进行相关版本的预测试)。

(5) 开发人员提交版本说明，包括该版本中新增加的功能特性、修改的缺陷、没有修改的缺陷、可能存在的问题以及测试重点的建议等。

如果开发团队的软件版本无法满足测试执行的入口准则，测试经理或者相关负责人可以拒绝进行测试，例如：测试对象无法通过预测试，测试经理可以拒绝开始测试执行活动。

4) 测试执行关键信息

要管理和监督测试的执行，测试人员应该在测试执行之前了解和掌握以下信息：

(1) 测试入口准则：指的是允许软件系统进入软件测试执行阶段所必须具备的条件，即软件系统满足什么样的条件，测试团队可以开始进行软件测试执行。测试执行的入口准则包括必须提交的文档、系统具备提交的条件、测试环境已经准备、测试工具可用等。

(2) 测试出口准则：指的是允许软件系统完成测试执行阶段所必须具备的条件，例如：所有的测试用例都被执行，并且所有严重程度为 1 和 2 的缺陷都已经关闭。

(3) 测试执行进度：测试计划中规定了测试执行的起始时间和结束时间，明确了测试人员每天应该完成的测试任务。

(4) 测试方法：测试计划中明确定义了测试采用的主要方法，例如：采用黑盒测试方法还是白盒测试方法。同时，测试方法中也定义了测试关注的主要测试类型，例如：是功能性测试还是稳定性测试、压力测试、兼容性测试等。

(5) 执行结果记录手段：记录手段规定了如何记录测试执行过程和结果。测试记录的手段可以是文档的方式，例如：Excel、Word，也可以是通过工具的支持。

(6) 测试监控：主要定义了如何监控测试执行的过程，以及责任人。另外也定义了监控的手段，例如：同行评审、阶段检查或随机检查等。

(7) 测试职责：指测试人员负责测试的领域和内容，以及相关的测试用例数据等。

(8) 测试用例：①测试用例设计的总体思路：测试用例设计的目的、测试的对象、测试判定的依据等；②测试用例选择：清楚了解测试人员负责的测试用例的内容；③测试用例执行的方式：测试用例的执行是通过手工运行方式来进行测试，还是自动化的测试方法。

(9) 测试环境：测试人员在测试执行准备的时候，还需要了解测试设计规格说明、测试用例规格说明和测试规程规格说明，掌握测试要求的环境以及如何搭建测试环境。测试用例执行对测试环境的要求，特别是自动化测试方式的时候需要什么样的测试自动化平台，采用什么样的运行方式等。

(10) 测试数据的准备：测试执行准备中，测试数据的准备是一个工作量很大而且技术要求很高的活动。因此，如何准备大量的测试数据，以及如何准备高质量的测试数据以满足测试的需求，这是测试执行准备活动一个非常重要的步骤。它主要包括测试数据的收集和优化。

5) 测试数据收集

测试数据的准备首先要进行测试数据的收集。主要的数据来自包括下面三种方式：

(1) 根据被测系统的需求分析，针对正常业务、异常情况、边界情况等来构建完整的数据。通过这种方式构建的数据不仅包括最基本的基础数据，例如：用户、权限、配置、基础编码、原始数据等，还包括软件系统在真实环境中需要用到的业务数据。这对于小型的软件系统来说还是可行的，但对于大型的系统来说可能是一个巨大工程。

(2) 利用现有系统，这适合已有类似系统的情况，测试是针对升级或者增加功能的系统。这种情况把已经在实际环境中运行的数据导出，在此基础上再进行数据的整理，最后生成测试数据。

(3) 将现有非电子化的业务数据录入到系统中，在验证业务的同时也完成了测试数据的积累，即在测试进行的同时进行数据的积累。但是这种情况下积累的数据往往有一定局限性，因为已经发生的业务数据基本是正确的、一致的，而且可能缺少某些特定业务的数据(不常发生的业务)。这样就需要根据对测试需求的分析，添加新的测试数据，以便能完整覆盖软件系统处理功能和业务能力。

6) 测试数据优化

数据来源确定后，还需要对收集的数据进行一系列的优化，从而保证数据的质量。数据的质量一般要满足测试需求、覆盖被测业务、覆盖测试边界以及要满足完整性、一致性等要求。测试数据的优化主要包括：

(1) 验证：验证数据的正确性、完整性、一致性以及判断是否满足测试的需求等。

(2) 修改：对错误的测试数据进行修正。有些数据(例如：状态、时间日期、模拟容错的数据等)需要根据测试的需要进行相应修改。

(3) 转换：对涉及安全和敏感的数据，例如：账户、密码、个人信息、身份证、联系方式等，需要进行适当的加密和转换。

(4) 补录：对比测试需求，补充符合测试需求的数据，例如：边界值、覆盖分支的等价类数据等。

(5) 清除：清除冗余的测试数据、错误的测试数据，以及由于产品的升级和更新已经不

适合测试需求的测试数据。

经过整理好的数据要纳入配置管理，以后根据变更进行数据的维护和更新，以保证满足系统测试的要求。

3. 测试执行的结束

测试结束活动中相关数据的分析主要用来回答下面这些问题：测试过程中哪些方面做得不够好、为什么会存在这些问题、哪些方面做得比较好以及积累的经验、如何在下一个项目中能够做得更好等。这些内容的分析可以让测试团队成员了解测试过程中的经验和教训，从而帮助测试团队在以后的测试中尽量避免重复以前的错误。同时，这些经验教训也可以帮助其他项目和其他项目团队改进他们的开发和测试过程，以及提高产品的质量。通过测试过程的评估(包括测试任务、所花费资源和所达到结果的鉴定评估)，可以发现哪些方面需要进一步改进。把这些发现结果使用在以后的项目中，可以帮助后续项目的持续改进。

当确定测试结束后，应当收集主要的输出成果，并且交给相应的人员归档，这些活动称为结束活动。测试结束活动主要包括以下四个方面：

1) 确保所有测试工作全部完成，例如：所有计划的测试都已经执行；提交的缺陷已经修改，并且进行了相应的确认测试和回归测试；遗留缺陷都经过项目团队的风险分析或相关资源的分析，认为在当前版本不进行该缺陷的修复而存在的风险是可以接受的，或者当前的资源限制无法解决这个缺陷，确定这些缺陷需要留到下一个版本解决。

2) 移交测试工作产品，例如：延期的或者无法解决的缺陷需要和使用产品的用户进行沟通；将测试文档和测试环境等移交给后续进行维护测试的小组。

3) 总结经验教训。记录开发过程和测试过程中所有的经验教训，并且将经验教训文档化，以免在以后的项目和项目测试中重复这些错误，例如：由于在测试的后期发现不曾预料的缺陷集群。测试团队分析之后发现，假如在早期的风险识别会议上邀请更加广泛的客户代表来参加，就可以减轻或者避免这类风险的发生。

实际的工作量和原来估算的工作量差距很大。分析工作量估算误差大的原因，例如：遗漏某些测试工作，在以后的工作量估算中，将这些因素考虑在内，不断提高估算的精度。

缺陷趋势的分析、引起缺陷的原因和影响的分析，例如：是不是由于项目后期的变更请求影响了开发的质量；是否由于采用了不好的实践，例如：裁减了某个测试级别，而这个测试级别可以较早地发现缺陷，从而提高测试效率、降低修改成本和节省测试时间；是否由于使用了新的技术、测试人员发生变动以及缺少相应项目的技能。

4) 在配置管理系统中归档所有的结果、记录、报表和其他文档及交付物，例如：测试计划和项目计划。

上述测试结束活动非常重要，而在实际测试过程中却常被遗漏。因此，应该将测试结束活动明确包含在测试计划中。测试结束活动的主要输入包括：

(1) 测试计划，例如：测试计划中的工作量估算、风险分析、采用的测试设计技术以及测试需求分析等。

(2) 测试用例完成数目，包括设计和执行的测试用例的数目，以及尚未执行的测试用例数目等。

(3) 缺陷数目，包括发现的缺陷数目、修复的缺陷数目、未修复的缺陷数目等。

(4) 缺陷的分布，包括缺陷发现趋势、缺陷修复趋势、缺陷在不同模块或者测试类型的分布等。

(5) 测试团队成员的周工作报告和月工作报告等。

测试结束活动的主要输出：测试经验教训总结、测试过程改进建议、归档的测试工作产品。

7.1.4　软件测试执行的控制

测试执行控制活动应该贯穿于整个测试执行过程，也包括软件执行过程。通过项目测试执行状态和其他的测试执行进度信息，和制定的测试计划进行比较，发现其中的偏差和问题，采取相应的手段来对测试执行活动进行控制，从而使测试执行活动能够按照测试计划进行。另外，测试执行监督和控制得到的反馈数据和信息也有助于修改和更新原先制定的测试执行计划。

测试执行控制并不是孤立活动，它受到很多因素的制约和影响。测试执行控制必须对测试执行提供的信息和发生的问题及时做出回应，同时做出相应计划的变更，例如：执行动态测试在某个原来认为质量较高的领域发现许多缺陷，或者由于测试开始时间延迟导致测试执行时间太短等，在这些情况下都需要对测试风险分析和测试计划进行相应的变更。变更也可能会要求对后续的测试执行工作重新进行优先级划分。

要管理和控制测试的执行，必须要有辅助工具。辅助工具一般是一款或几款软件管理软件，可以管理测试用例、自动化测试脚本、运行自动化脚本的通过和失败记录、报告发现的缺陷，追踪缺陷、测试执行进度等等。惠普的 ALM 等软件有很丰富和实用的功能可以支持测试执行的管理和控制。在本书的实践部分会专门介绍并动手实践。本节不做具体介绍。

前面的课程内容已经多次说明了制定软件测试计划是软件测试整体生命周期中最基本和重要的一个环节。测试执行控制也需要特别注意在测试的执行过程中评估、分析和改进及有的测试计划、设计的测试用例，以便使软件测试计划和设计的测试用例达到优化。例如：从测试执行活动中得到的反馈信息可以识别实际测试中存在的新风险，从而需要对测试计划中的风险部分内容做相应修改。下面加以说明和分析：

总之，测试执行控制阶段的主要测试活动有：

(1) 按预定的计划执行测试：熟悉和理解测试计划，按预定的目标执行测试计划、运行自动化测试脚本和执行测试用例。测试人员在执行测试时根据测试计划中分配给自己的测试任务和提供的测试用例，具体运行相应的测试用例，并验证预期的结果。同时根据需要记录实施用例的结果，提交测试记录。发现缺陷时提交缺陷报告。

(2) 确定测试执行范围和风险：确定测试执行的主要范围，测试质量的风险，并且根据风险分析的结果来确定测试执行的重点和优先级。

(3) 确定测试执行目的：不同组织、不同项目以及不同的测试级别，其测试执行目的可能是不一样的。测试执行目的可以是在研发的后期，发布之前的一次全面的测试走查(Test Pass)证明软件系统是可以正常工作的，也可以是尽可能多地发现缺陷，也可能是通过收集测试信息和数据，用来改进软件开发和测试过程，或者上面这些测试目的的兼而有之。因此，需要

在测试计划阶段明确测试执行的目的，从而采用不同的测试方法，分配相应的测试执行资源。

(4) 确定测试执行方法：针对测试执行目的和测试级别的不同，需要确定不同的测试执行技术和测试方法，例如：静态测试(评审和静态分析)在软件开发生命周期的早期使用得比较多。测试执行期间，运行所有自动化测试脚本，每天发布运行结果，即自动化脚本通过率，报告未通过的脚本出错信息，发邮件给相关的测试人员以及时调查失败原因，遇到缺陷及时报告和沟通。

(5) 确定测试执行资源：测试计划中规定的测试所需的资源对测试执行成功非常关键。测试执行资源包括测试的人力资源、测试执行使用的 PC 和相关的测试管理工具、测试所需的网络设备、测试仪表以及测试环境的分配等。

(6) 计划测试执行的进度：根据整个软件项目的进度和测试执行范围，安排整个测试执行活动的进度，以及为每个测试执行活动分配相关的人力资源。

(7) 确定测试执行入口准则和出口准则：在测试计划中需要确定测试执行入口准则、测试执行出口准则、测试挂起准则和测试恢复准则。

(8) 监控和记录测试执行过程：监控测试执行过程的信息包括计划测试进度和实际测试进度的比较、测试覆盖率和测试出口准则、测试挂起准则的满足程度等。

(9) 度量和分析测试结果：根据测试执行过程得到的度量数据(例如：测试执行发现的缺陷数目、测试用例执行的数目、测试用例失败的数目等)对测试执行过程、测试质量和产品质量进行分析，并根据分析结果采取合适的措施和应对计划，例如：测试重点和优先级的调整。

(10) 修正测试执行计划：根据前面测度和分析的结果，以及测试执行监控过程中得到的信息，根据需要对测试执行相关活动进行修正。

(11) 做出决定：根据测度和分析的结果做出相应的决定，包括重新启动测试执行、测试执行结束、测试挂起、测试相关的变更、测试应对计划实施等。在实际测试执行过程中，测试执行是否可以继续或者中断、结束通常还需要考虑时间和成本方面的因素。如果这些因素导致强制停止测试执行活动，则可能是因为在项目计划中没有配置足够的测试资源或总体经费提前用完，或者低估了某项测试执行活动的工作量。

测试监视旨在为测试控制提供反馈信息和可视性。监视的信息可通过手工或自动方式进行收集，同时，这些信息可以用来衡量测试计划中定义的出口准则(例如：测试覆盖率)，也可以用度量数据对照测试计划的时间进度和预算来评估测试的进度。常用的度量指标有：

(1) 在测试分析和设计中发现的缺陷数。

(2) 测试用例设计完成率。

(3) 测试环境准备的进度。

(4) 测试用例执行情况(例如：测试用例执行率、测试用例通过率)。

(5) 缺陷信息(例如：缺陷密度、发现和修改的缺陷比例、再测试的通过率)。

(6) 需求、风险或代码的测试覆盖率。

(7) 测试的成本。

通过测试监视活动，可以掌握测试活动的时间进度、资源、成本和产品质量等信息，这些信息将有利于在测试控制活动中做出正确的决定。

测试执行控制活动要根据收集和报告的测试信息来采取相应的应对措施。措施可以针对任何测试活动，也可以包括任何软件开发生命周期中其他的活动。下面通过一些例子来说明测试执行控制活动是如何开展的。

如果在测试执行过程中的被测试版本在基本功能方面经常发现缺陷，导致很长一段时间内大量的测试用例无法执行，将严重影响测试执行进度。这种情况下，可以采用不同的策略进行应对，例如：可以根据挂起准则挂起本次测试执行；或者在测试团队对版本正式进行测试执行之前，首先进行冒烟测试(预测试)，组成冒烟测试的测试用例可以由测试团队和开发团队共同挑选，确保它们覆盖了基本功能，每次被测试版本提交测试团队之前，必须保证这些预测试的测试用例全部通过。由于组成预测试的测试用例是从原来的测试集中挑选出来的，所以预测试通过后，对应的一些测试用例就可以直接标识为测试通过，避免同样的测试用例在测试过程中重复执行。

如果在测试执行过程中发现大部分缺陷是功能性的，而不是非功能性的(例如：兼容性、性能、可移植性等)。这时就需要对执行的测试用例进行分析：是因为软件版本质量不好，还是因为缺乏相应的非功能性测试用例。如果是由于测试用例不完善造成的，则需要对测试用例进行补充。

测试执行阶段的主要测试活动包括测试数据配置的准备、测试环境的搭建、测试结果的比较、测试日志记录和回归测试等。为了有效地对测试实现和执行阶段进行监控，可能采用的度量包括：

(1) 测试环境配置的百分比。
(2) 测试数据装载的百分比。
(3) 测试条件和测试用例执行的百分比。
(4) 测试用例自动化的百分比。

监控测试进度的度量可以和测试出口准则联系起来(在测试计划中确定)。评估出口准则和报告阶段涉及的度量有：

(1) 测试需求的覆盖率。
(2) 测试用例的覆盖率。
(3) 测试用例执行通过/失败的数目。
(4) 提交的缺陷数目，根据缺陷的严重程度和优先级进行的分类。
(5) 提交的缺陷数目，接受的缺陷和被拒绝的缺陷的比例。
(6) 计划成本支出和实际成本支出的偏差。
(7) 计划花费时间和实际花费时间的偏差。
(8) 测试中识别的风险和处理的风险数目。
(9) 由于事件制约因素浪费的时间。

7.2　软件测试执行结果的评估

传统上定义测试出口准则也是测试计划中的一个重要内容，其目的是确定什么时候可以

结束测试。当提交的测试对象满足测试执行的入口准则时，就可以开始测试执行了。测试执行通常是依据测试规程规格说明展开的。测试用例的执行既可以通过手动方式进行，也可以通过自动方式进行。对手动测试来说，测试执行主要是测试人员根据测试用例中的步骤进行测试执行，包括测试人员输入必要的测试数据、检查测试输出、比较测试实际结果和期望结果、记录在测试过程中发现的问题等。而对于自动化测试过程，测试执行可能是利用测试工具运行测试脚本，通过自动化方式记录测试结果。例如：判断某个测试级别是否可以结束，需要判断当前的测试是否达到了规定的目标。测试出口准则可以基于下面的一些度量进行判断：

1) 测试完整性度量，例如：代码、功能或风险的测试覆盖率。

2) 对软件缺陷密度、缺陷趋势或系统可靠性的度量。

3) 继续测试成本度量，评估继续测试的收益和结束测试存在的风险之间的平衡关系。

4) 可能遗留的风险的度量，例如：评估没有被修改的缺陷、在某些区域测试覆盖率较低等对项目或者产品可能产生的风险。

5) 项目产品交付时间的度量，例如：客户要求的软件产品交付时间或者测试计划规定的截止时间等。

6) 测试组长根据此轮测试的结果，编写阶段性测试报告(参考测试阶段性报告模板)，主要应包含以下内容：

(1) 测试报告的版本；

(2) 测试的人员和时间；

(3) 测试所覆盖的缺陷(测试组在这轮测试中所有处理的缺陷，报告测试组长处理的缺陷和实施工程师验证的缺陷。不仅要写出覆盖缺陷的总数，还要写明这些缺陷的去向)；

(4) 测试新发现的缺陷数量；

(5) 上一版本活动缺陷的数量；

(6) 经过此轮测试，所有活动缺陷的数量及其状态分类；

(7) 测试评估：写明在这一版本中，那些功能被实现，那些还没有实现，这里只需写明和上一版本不同之处即可；

(8) 亟待解决的问题：写明当前项目组中面临的最优先的问题，可以重复提出。

7) 测试组长根据测试的结果，按照测试总结报告的文档模板编写测试报告(参考测试总结报告模板)，测试报告必须包含以下重要内容：

(1) 测试资源概述：多少人、多长时间；

(2) 测试结果摘要：分别描述各个测试需求的测试结果，产品实现了哪些功能点，哪些还没有实现；

(3) 缺陷分析：按照缺陷的属性分类进行分析；

(4) 测试需求覆盖率：原先列举的测试需求的测试覆盖率，可能一部分测试需求因为资源和优先级的因素没有进行测试，那么在这里要进行说明；

(5) 测试评估：从总体对项目质量进行评估；

(6) 测试组建议：从测试组的角度为项目组提出工作建议。

7.2.1　测试通过与失败

测试结果的比较是测试用例执行过程中一个很重要的环节。必须仔细检查每个测试用例执行的实际输出结果，根据测试期望结果判断被测系统是否能够正确地工作。测试结果包括中间测试结果和最终测试结果，因此，在测试过程中不仅要对最终结果进行比较，也需要对中间结果进行比较。测试执行对每一项要测试的内容都必须有个结论。即测试是否通过。答案是“是(Yes)”或者“否(No)”。

英文通过考试是叫 Pass a test。考试过关就是按照预定的标准，通过了验证。对于软件测试领域，英文 Test Pass 是按照预定的软件测试标准，经测试人员验证后认为该项测试要验证的功能或设计满足需求标准，被认为是通过了。实际工作中，Test Pass 常作名词用，指把所有要测试的案例(自动化的和非自动化的)以及其他需要包括其中的测试活动在规定时间范围内，整个运行一遍的一种测试活动。

假如测试实际输出结果和测试期望结果一致，那么认为该测试用例执行是通过的；否则就认为是测试失败，也就是测试未通过。只要和预期测试结果不一致，需要进一步检查以确定引起不一致的原因。 测试结果的不一致或者失败并不一定是由于测试对象的缺陷引起的，也许是应为测试环境出错、测试人员执行测试时人为误差等。例如：由于测试用例设计存在问题、测试用例的期望结果存在问题、测试环境存在的问题等。假如是由于测试对象引起的不一致，那么测试人员需要提交相应的缺陷报告。所以，在比较测试结果的时候，需要测试人员从不同的方面来确认具体的问题来源。

测试结果的比较手段可以是手动比较，也可以是工具自动比较。手动比较由测试执行人员将测试用例的执行结果和测试用例的期望结果进行比较。手动比较的优点是灵活(不需要固定的描述格式)和自由(可随时选择所需要的测试用例加以执行)；其缺点是效率低、容易出错。自动比较由自动化管理系统(例如：自动化测试工具)自动完成测试实际结果和测试期望结果比较的任务。自动比较要求系统对测试用例(或数据)有明确的描述方法和比较机制。自动比较的优点是效率高和准确性高；它的缺点是灵活度和自由度受测试工具和运行自动化测试脚本测试环境的影响。

7.2.2　测试覆盖率与通过率

在测试分析和设计阶段，最主要的测试活动是识别测试范围、测试目的和设计测试用例，为更好地对这个阶段的测试活动进行有效地监控，需要定义合适的测试度量指标。在 CMMI4 体系的测试过程中定义了四个度量指标：测试覆盖率、测试执行率、测试执行通过率、测试缺陷解决率。测试执行人员应该正确理解这四个度量指标。

本节先分析测试的覆盖率，进而说明测试执行的通过率。因为两者是相辅相成的。如果测试的覆盖率很低，测试通过率高也不意味着达到测试执行的满意效果。最佳的结果是有最高的测试覆盖率和最高的测试通过率。

1. 测试覆盖率

测试覆盖率是用来度量测试完整性的一个指标。测试覆盖率从纬度上说包括广度覆盖和

深度覆盖；从内容上说包括用户场景覆盖、功能覆盖、功能组合覆盖、系统场景覆盖。

首先说广度，是否需求规格说明书中的每个需求项都在测试用例中得到要验证的覆盖。其次说深度，通俗地说，是不是我们的测试设计流于表面，是否能够透过客户需求文档，挖掘出可能存在问题的地方。例如：重复单击某个按钮 10 次，或者依次执行新增、删除、新增同一数据的记录、再次删除该记录操作。

在设计测试用例时，测试人员可能不会常考虑单独设计广度或深度方面的测试用例，而一般是结合在一起设计。为了从广度和深度上覆盖测试用例，我们需要考虑设计各种测试用例，如：用户场景(识别最常用的 20%的操作)、功能点、功能组合、系统场景、性能、语句、分支等。在执行时，需要根据测试时间的充裕程度按照一定的顺序执行。通常是先执行用户场景的测试用例，然后再执行具体功能点、功能组合的测试。

覆盖率可以通过一个比率公式来表示：

覆盖率 = (至少被执行一次的 Item 数)/Item 的总数

侧重测试用例的覆盖率=已设计测试用例的需求数/需求总数。

通过覆盖率数据，可以知道测试是否充分，测试的弱点在哪些方面，进而指导设计能够增加覆盖率的测试用例和测试实践。测试的覆盖率有不同的类型和侧重。

测试用例应该对系统需求进行覆盖。通常来讲，测试用例对系统需求的覆盖率要达到 100%。当测试用例设计完成后，假如测试用例的需求覆盖率没有达到 100%的要求，就需要增加测试用例来覆盖遗漏的需求，以保证测试用例对需求的覆盖率达到 100%。

比如，以测试用例覆盖率分析：某待测试软件功能有 60%测试用例实现了自动化，这些测试用例由自动化测试管理系统和专人负责运行，并把运行结果报告给测试执行人员。剩下的 40%测试用例需要手动测试。其中 60%自动化测试覆盖的部分运行按照计划而且通过率是 100%，而 40%手动测试应执行的部分因人员流动和其他资源未满足而只完成了 20%。那么该功能执行测试用例的总体测试覆盖率约为 80%。虽然自动化测试部分的覆盖率是 100%完成。

2. 测试通过率

测试通过率是用来度量测试执行结果的一个指标。执行率常指实际执行过程中确定已经执行的测试用例比率。

计算公式 = 已执行的测试用例数/设计的总测试用例数

在实际测试过程中，经常有如下这种情况发生：一种情况是，因为系统采用迭代方式开发，每次构建时都有不同的重点，包含不同的内容；第二种情况是，由于测试资源的有限，不可能每次将所有设计的测试内容都全部测试完毕。由于这两种情况的存在，所以在每次执行测试时，我们会按照不同的测试重点和测试内容来安排测试活动，所以就存在了“测试执行率”这个指标。

通常，我们的测试目标是确保 100%的测试用例都得到执行，即执行率为 100%。但是，如前面所提到的，实际中可能存在非 100%的执行率。如果不能达到 100%的测试执行率，那么我们需要根据不同的情况制定不同的测试执行率标准——主要考虑风险、重要性、可接受

的测试执行率。在考虑可接受的测试执行率时，就涉及测试用例执行顺序的问题。

在设计测试用例时，我们需要从广度和深度上尽可能的覆盖需求，所以我们就需要设计各种测试用例，如正常的测试用例、异常的测试用例、界面的测试用例等。但是在执行时，测试人员需要根据项目进度和测试时间的充裕程度，参考测试执行率标准，将测试用例按照一定的顺序执行。通常是先执行用户场景对应的测试用例，然后执行具体功能点、功能组合的测试，完成这些测试后，再进行其他测试，如系统场景、性能、语句等测试。

测试执行通过率指在实际执行测试用例中，执行结果常指“通过”的测试用例比率。

计算公式 = 执行结果为“通过”的测试用例数/实际执行的测试用例总数。

可以针对所有计划执行的测试用例进行衡量，可以细化到具体模块，用于对比各个模块的测试用例执行情况。

为了得到测试执行通过率数据，在测试执行时，需要记录下每个测试用例的执行结果，然后在当前版本执行完毕，或者定期(如每周)统计当前测试执行数据。通过原始数据的记录与统计，就可以快速得到当前版本或当前阶段的测试执行通过率。

3. 缺陷解决率

缺陷解决率，指某个阶段已关闭缺陷占缺陷总数的比率。缺陷关闭操作包括以下两种情况：

正常关闭：缺陷已修复，且经过测试人员验证通过；

强制关闭：重复的缺陷；由于外部原因造成的缺陷；暂时不处理的缺陷；无效的缺陷。这类缺陷经过确认后，可以强制关闭。

计算公式 = 已关闭的缺陷/缺陷总数

在项目过程中，在开始时缺陷解决率上升很缓慢，随着测试工作的开展，缺陷解决率逐步上升，在版本发布前，缺陷解决率将趋于 100%。一般来说，在每个版本对外发布时，缺陷解决率都应该达到 100%。也就是说，除了已修复的缺陷需要进行验证外，其他需要强制关闭的缺陷必须经过确认，且有对应的应对措施。可以将缺陷解决率作为测试结束和版本发布的一个标准。如果有部分缺陷仍处于打开或已处理状态，那么原则上来说，该版本是不允许发布的。

缺陷关闭数据，可以通过缺陷跟踪工具定期(如每周)收集当前系统的缺陷数、已关闭缺陷数，通过这两个数据，即可绘制出整个项目过程或某个阶段的缺陷解决率曲线。

当然，测试度量指标不仅包括上述四个，在实际工作中，还会用到诸如验证不通过率、缺陷密度等指标。收集这些数据目的是为了能对测试过程进行量化管理。但是，简单收集度量数据不是目的，通过对数据的分析、预防问题、对问题采取纠正措施，减少风险才是目的。

7.2.3　测试通过标准

测试中使用的出口准则可以用于报告和计划什么时候可以停止测试。出口准则(Exit Criteria)定义是与利益相关者达成一致的系列通用和专门的条件，用于正式定义一个过程的结束点。出口准则的目的可以防止将没有完成的任务错误地看成任务已经完成。

软件测试过程中的测试进度监控可以收集合适的测试相关信息，来支持和评估测试出口

准则，例如：通过度量测试执行的完成率来判断测试执行的进度。测试评估可以从测试过程和测试质量两个方面进行。通过对测试的评估，可以量化测试过程，判断测试进行的状态，以及判断测试结束的时间。同时，通过测试评估，可以生成一些量化数据，例如：测试覆盖率、缺陷清除率、测试完成率等，它们可以作为软件质量评估的输入。再比如，测试通过率为：功能性测试用例通过率达到 100%；非功能性测试用例通过率达到 80%时。

测试评估可以是软件测试的一个阶段性结论，通过生成的测试评估报告来确定测试是否达到出口准则。测试评估应该贯穿于整个软件测试过程，可以在每个测试阶段结束前进行，也可以在测试过程的某个时间点上进行。从测试过程的角度而言，测试进度控制可以收集正确的测试信息，用来帮助提交测试的状态报告，包括测量测试完成的进度。

评估测试出口准则和报告阶段的主要测试活动有：

1) 将测试状态和测试计划中的出口准则进行比较。

2) 评估是否需要更多的测试执行，或者是否需要更改测试出口准则。

3) 输出测试总结报告。

评估测试出口准则和报告阶段的主要输入有：

1) 测试状态报告、缺陷状态报告、风险状态报告、项目测试周报告/月报告、测试出口准则和测试计划。

2) 回归测试所运行的用例全部通过。

3) 缺陷经过验证。

4) 所有缺陷都被指明处理方式。

5) 同行审查没有新的缺陷或没有严重缺陷产生。

进行对测试组所测试项目或产品的测试审查工作。基本原则为：

1) 不依据所设计测试用例，进行自由测试。

2) 测试时间保持在 3 个正常工作日以内。

3) 如发现严重缺陷，则一轮测试结束后更新版本并执行回归测试。

4) 提交当日测试记录。

5) 编写同行审查总结报告(报告以简单为好)。

测试出口准则的评估是检验测试对象是否符合预先定义的一组测试目标和出口准则的活动。测试出口准则的评估可能产生下列结果：测试结果满足所有的出口准则，测试活动可以正常结束；可能会要求执行一些附加测试用例；或者测试出口准则要求过高，需要对测试出口准则进行修改。当进行测试出口准则评估的时候，测试人员必须决定当前测试状态是否完全满足测试计划中的测试出口准则，例如：测试对象有 80%的语句被执行，可作为测试出口准则的条目之一。

如果执行完所有的测试用例后，测试出口准则的一个或多个条目还没有满足，那么应当执行更进一步的测试或者修改测试出口准则。如果需要增加测试用例，需要注意保证新的测试用例满足相应的出口准则。否则，额外的测试用例只会增加工作量，而不会对测试结果有任何改进。

为了满足出口准则，有时需要采用不同的测试技术，例如：在测试系统对某异常情形响

应时，由于现行的测试环境不能够引入或者模拟这种异常情形，处理这种异常情形的代码就不能够被执行和测试。这种情况下，应当使用其他的测试方法(例如：静态分析)对代码进行分析和评估。

另一种可能情况就是由于被测试对象本身的问题，使得定义的出口准则无法满足，例如：测试对象中包含死代码，导致无法满足 100%的语句覆盖率。在评估测试出口准则的过程中，要对诸如此类的可能性进行考虑，避免因为无意义的测试出口准则而进行无效的测试。前面例子中出现出口准则条目无法满足的时候，应该调查为什么测试对象会包含死代码，这样做可能发现更多缺陷。

即使在测试过程中消耗比测试计划更多的资源，但是由于更多地消除了测试对象中的缺陷，测试也在一定程度上降低了软件的质量风险。通常来说，测试对象中的缺陷导致测试对象在运行过程中出现失败而引起的成本，远远高于在测试时发现并修复缺陷的成本。

完成相关的测试活动后，测试团队需要向项目利益相关者提交测试报告。测试报告中需要明确是否满足测试出口准则，或者罗列尚未满足出口准则的具体条目。不同的测试级别，其输出的测试报告形式可能是不一样的，例如：对于组件测试这样低级别的测试，测试报告的形式可能仅是向项目经理汇报关于是否达到出口准则的一个简单信息；而在相对高级别的测试中(例如：系统测试)，可能需要输出正式的测试报告。

根据第 5 章定义的缺陷分类的方法，测试执行的通过标准可以用以表 7-1 形式注明。

表 7-1　测试执行的通过标准

A 类错误	B 类错误	C 类错误	D 类错误	E 类建议
无	无	≤2%	≤4%	暂不作要求

7.2.4　测试执行结果报告

所有的测试执行活动完成并输出测试报告后，并不意味着测试活动全部结束。测试经理和测试团队中的其他成员还需要将测试工作产品归档，同时对测试过程和测试活动进行相关数据的收集和分析，总结测试过程和测试活动的经验教训，例如：测试活动是否实现了测试计划设定的目标、有哪些意外事情和风险发生、发生的原因是什么、是否有效地解决了这些风险、是否存在没有解决的变更请求等。例如：失败发现率。如果失败发现率降到给定的阈值(例如：失败发现率小于 0.2 个/人天)，就表明不再需要更多的测试，测试工作可以结束。根据失败发现率评估测试出口准则时还应该考虑失败的严重程度，对失败进行分类，并区别对待。

测试执行结果报告是必须要有的一个环节，因为测试负责人以及整个软件开发项目负责人需要了解软件测试执行后的综合结果。最后要出一个报告。对不同的软件产品、不同的企业、项目团队，其测试报告的模板和内容有很多选择。本节列举一个实例，以说明常见的测试执行结果要考虑的内容和用图形和数据直观表述测试执行结果的方法。

1. 一个测试执行的结果报告模板(XX 开发项目-测试执行结果报告)

1) 概述

2) 编写目的

(1) 通过对测试结果的分析得到对软件的评价；

(2) 为纠正软件缺陷提供依据；

(3) 使用户对系统运行建立信心。

3) 参考资料

说明软件测试所需的资料(需求分析、设计规范等)。

4) 术语和缩写词

说明本次测试所涉及的专业术语和缩写词等。

5) 测试对象

包括测试项目、测试类型、测试批次(本测试类型的第几次测试)、测试时间等。

6) 测试分析

(1) 测试结果分析：列出测试结果分析记录，并按下列模板产生缺陷分布表和缺陷分布图。分析模版：①从软件测试中发现的并最终确认的错误点等级数量来评估；②从以上提出的缺陷等级来统计等级和数量的一个分布情况，如表 7-2 及图 7-3 所示。

表 7-2　缺陷严重级分布表

严重度等级	A = 非常严重	B = 很严重	C = 严重	D = 一般	E = 轻微
缺陷数量	2	17	3	0	1
所占比例	9%	74%	13%	0%	4%

(1) 对比分析。若非首次测试，将本次测试结果与首次测试、前一次测试的结果进行对比分析比较。

(2) 测试评估。通过对测试结果的分析提出一个对软件能力的全面分析，需标明遗留缺陷、局限性和软件的约束限制等，并提出改进建议。

(3) 测试结论。根据测试标准及测试结果，判定软件能否通过测试。

图 7-3　缺陷按严重级分布示意图

7) 测试主管签字：　　　　　　　年　　月　　日

2. 缺陷状态报表

团队开始执行测试并报告和修复缺陷后，可以通过生成和查看“缺陷状态”报表来跟踪团队在解决和关闭缺陷方面的进度。 此报表基于缺陷状态、优先级别和严重级别显示累计缺陷计数。参见表 7-3。

1) 此报表可用于回答以下问题：

(1) 团队修复缺陷的速度是否可以按时完成工作？

(2) 团队是否先修复高优先级别缺陷？

(3) 缺陷按优先级别和严重级别的分发情况如何？

(4) 为每个团队成员指派了多少缺陷？

表 7-3　缺陷状态报表

缺陷数	所有缺陷的累计计数的可视表示形式，按其状态分组
活动缺陷(按优先级别/严重级别)	描述仍处于活动状态的缺陷数的饼图，按优先级别或严重级别分组
活动缺陷(按指派)	水平条形图，其中包含每个团队成员分配给活动状态的缺陷的总数，按优先级别或严重级别分组
已解决的缺陷(按指派)	水平条形图，其中包含每个团队成员分配给已解决状态的缺陷的总数，按优先级别或严重级别分组

2) 为使“缺陷状态”报表更有实用价值，团队可以执行以下活动：

(1) 定义缺陷，然后指定其“迭代”和“区域”路径。

(2) 指定每个缺陷的“优先级别”和“严重级别”。

(3) 将每个缺陷指派给负责解决或关闭缺陷的团队成员。

(4) 随着每个缺陷的修复、验证和关闭，更新其状态。

3) 要了解团队在当前迭代中的进度，报表的开始和结束日期必须与当前迭代周期的开始和结束日期相符。

(1) 统计当前迭代周期的开始后状态处于活动或者新发现的缺陷。

(2) 将这些缺陷的状态变化进行统计，生成报表。

4) 在实际项目中，“缺陷状态”报表随你在产品开发周期中所处的阶段而异。早期迭代显示的活动缺陷数应逐步递增。临近开发周期末尾的迭代应显示大量已解决的缺陷。查看该报表可以确定在迭代或一段时间内的进度。具体而言，你可找到以下问题的答案：

(1) 团队解决并关闭缺陷的速度如何？

(2) 团队修复缺陷的速度是否足够快，可以按时完成工作？

(3) 团队是否先修复高优先级别缺陷？

(4) 缺陷按优先级别和严重级别的分发情况如何？

(5) 为每个团队成员指派了多少缺陷？

(6) 是否有任何团队成员在解决或关闭缺陷时需要帮助？

一般项目开发进展为正常状态的“缺陷状态”报表显示的活动如图 7-4 所示。缺陷数应

随时间递增，并且解决和关闭缺陷的进度是稳定的。如果团队修复的缺陷数多于发现的缺陷数，则活动的(Active)缺陷数会减少。

一般项目开发进展非正常状态的“缺陷状态”报表显示的活动如图 7-5 所示。

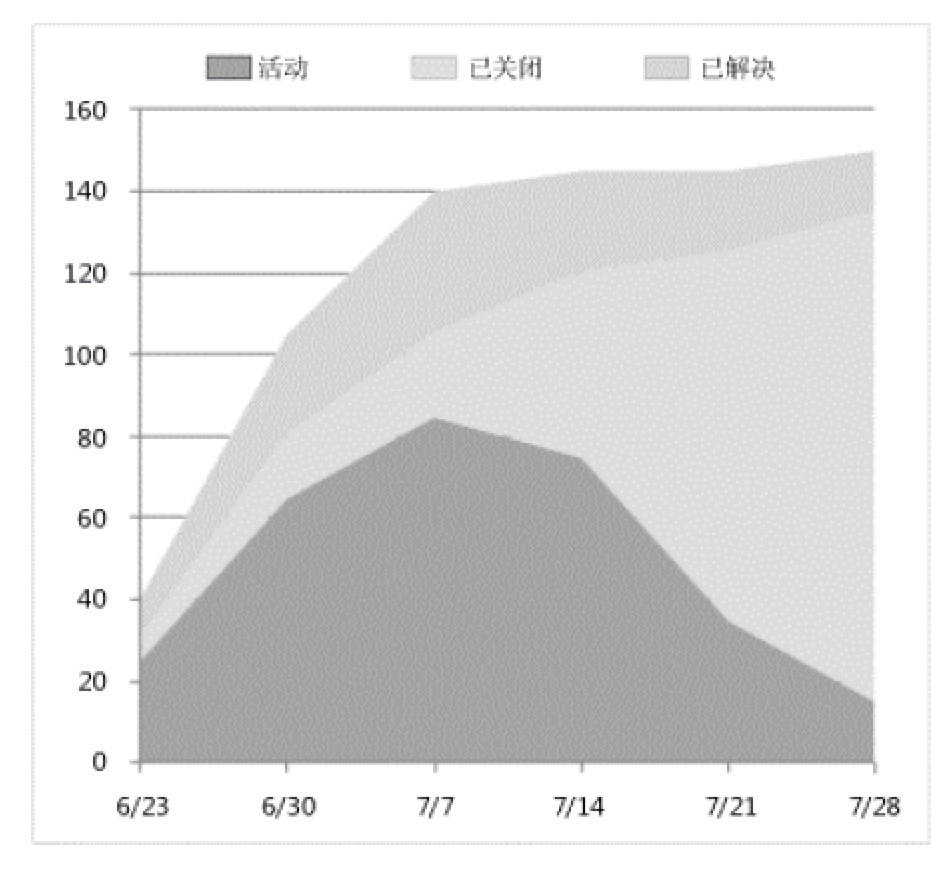



图 7-4 项目正常状态的“缺陷状态”图

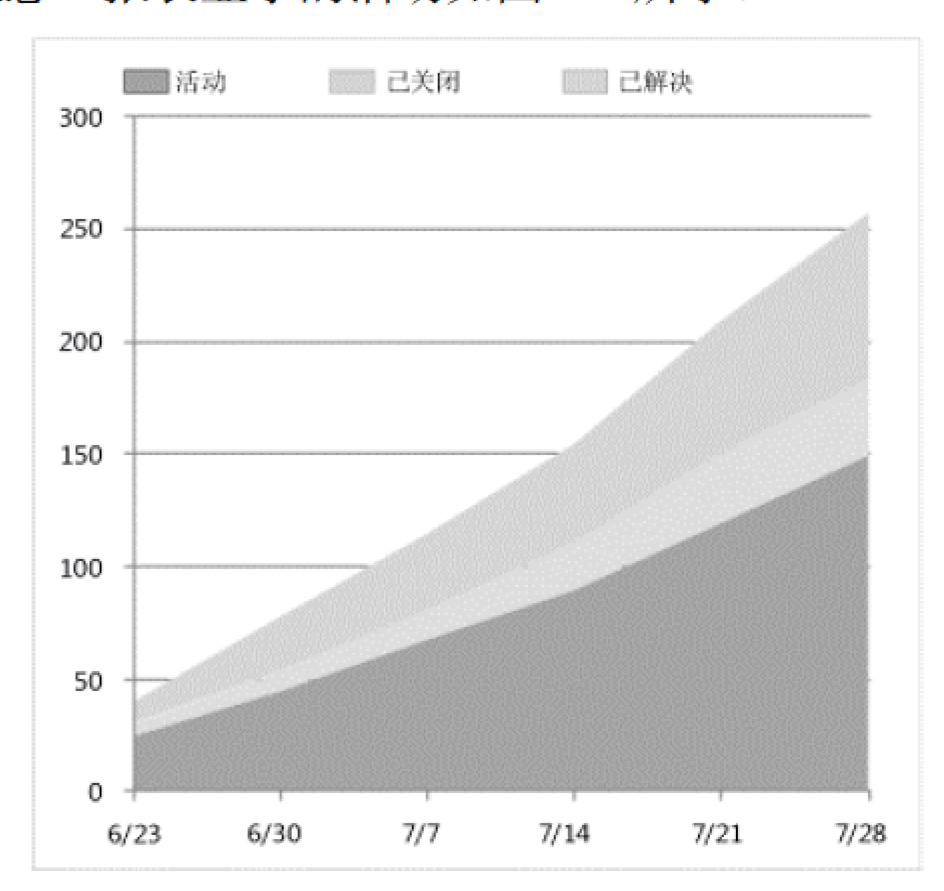


图 7-5 项目非正常状态的“缺陷状态”图

非正常的“缺陷状态”报表会显示下表所述的一个或多个指示性特征。参见表 7-4。

表 7-4 项目非正常状态的“缺陷状态”图分析

特 征	可能的问题
表示活动缺陷的带区变得越来越宽。如果表示活动缺陷的团队带区在变宽，则缺陷积压工作也在增加。团队发现的缺陷数量超出了其解决或关闭缺陷的能力。 变宽的活动缺陷带区可能表示瓶颈对团队解决和关闭缺陷的能力造成了阻碍	是否将团队成员重新分配给了其他非优先级别任务？ 是否有其他问题对团队解决和修复缺陷的能力造成了阻碍？
活动缺陷的数量没有变化。活动缺陷数量的平缓趋势表示团队没有在找缺陷	测试覆盖率是否足够？ 是否有其他问题对团队发现缺陷的能力造成了阻碍？
已解决或关闭的缺陷的数量没有变化。如果正在解决或关闭的缺陷的数量长时间保持不变，则表明团队成员可能无法解决或关闭缺陷	是否正确设置了团队优先级别？ 是否在其他任务上过度分配了团队成员？ 团队成员是否正确跟踪了其缺陷状态？
缺陷指派分布不平均。如果为一两个团队成员指派了大量缺陷，而为其他团队成员则只指派了少量缺陷，则团队可能要考虑重新指派工作	团队是否需要通过重新指派缺陷来平衡工作负荷？
活动高优先级别缺陷的数量大于活动低优先级别缺陷的数量。当高优先级别缺陷的数量比低优先级别缺陷的数量多很多时，可能会先致力于解决低优先级别的项	团队是否按团队设置的优先级别顺序修复缺陷？ 是否有问题对团队修复高优先级别缺陷的能力造成了阻碍？

5) 可以通过以下方式筛选“缺陷状态”报表或更改其显示：

(1) 更改报表的开始和结束日期。

(2) 通过更改迭代路径、区域路径、状态、严重级别或优先级别，筛选并显示计入的缺陷集。

(3) 基于优先级别或严重级别更改分组。

图 7-6 显示了用其他缺陷管理系统时可用的筛选器和显示选项：



图 7-6　缺陷管理系统可用的筛选器和显示选项

3. 验收测试结果报告(参见表 7-5)

表 7-5　测试执行结果报告

测 试 项 目		测 试 说 明	测 试 结 果
功能测试	功能挂接	各功能挂接正确	通过
	软件功能实现	××	通过
安全可靠性	安全保密性	具有用户身份验证及访问权限管理功能，安全性基本得到保障	通过
	软件容错性	软件发现错误时，有错误提示，并能恢复到正常状态	通过
	运行稳定性	在运行中没有因出错导致出现系统推出和死机现象	通过
用户界面	界面输入	界面输入正常	通过
	界面显示	在 1366×768 的分辨率下，界面显示正常	通过
	界面文字与提示	界面文字表示清晰，无错误和模糊定义	通过
中文符合性	界面中文符合性	界面使用统一的简体中文	通过
用户文档	用户文档完整性	用户文档的描述基本完整	通过
	用户文档正确性	用户文档的信息描述正确，无歧义	通过
	用户文档一致性	用户文档的描述与程序及数据一致	通过
病毒检测	病毒检测	运用杀毒软件进行检测，没有发现病毒	通过

技术相关文档：

1) 检查是否提交系统源代码；
2) 检查系统源代码书是否完整，且准确无误；
3) 检查是否提交安装盘；
4) 检查安装盘是否完整，且准确无误；
5) 检查是否提交需求说明书；
6) 检查需求说明书是否完整，且准确无误；
7) 检查是否提交概要设计说明书；
8) 检查概要设计说明书是否完整，且准确无误；
9) 检查是否提交详细设计说明书；
10) 检查详细设计说明书是否完整，且准确无误；
11) 检查是否提交数据库详细设计；
12) 检查数据库详细设计是否完整，且准确无误。

表 7-6 列出了测试执行结果报告。

表 7-6　测试执行结果报告

测 试 用 例	测 试 结 果
1) 是否提交系统源代码	√是/□否
2) 系统源代码是否完整	√是/□否
3) 系统源代码是否准确	√是/□否
4) 是否提交安装盘	√是/□否
5) 安装盘是否完整	√是/□否
6) 安装盘是否准确	√是/□否
7) 是否提交需求说明书	√是/□否
8) 需求说明书是否完整	√是/□否
9) 需求说明书是否准确	√是/□否
10) 是否提交概要设计说明书	√是/□否
11) 概要设计说明书是否完整	√是/□否
12) 概要设计说明书是否准确	√是/□否
13) 是否提交详细设计说明书	√是/□否
14) 详细设计说明书是否完整	√是/□否
15) 详细设计说明书是否准确	√是/□否
16) 是否提交数据库详细设计	√是/□否
17) 数据库详细设计是否完整	√是/□否
18) 数据库详细设计是否准确	√是/□否

7.3　软件测试执行的最佳实践

测试执行的需求、目标、范围、结果和流程等都会因不同的开发软件产品或项目的所处条件而有很多不同。但追求和推广最佳实践是业界一直在分享和鼓励的，也是测试执行管理者提倡和支持的。本节通过几个具体测试执行的环节来说明软件测试执行的最佳实践。

7.3.1　测试执行总结报告

测试执行总结报告是将数据收集和分析结果进行文档化，并且提交给相应的团队作为以后项目的参考。测试执行总结报告是进行软件测试过程评估和改进的重要输入，也是进行相关开发过程改进和测试度量数据库更新的主要输入。测试执行总结报告主要包括概要信息、测试风险、测试工作量、测试执行、测试缺陷和经验教训等。其中几项值得注意的部分特别加以说明如下。

1. 概要信息

测试执行总结报告首先需要简单介绍测试执行相关的信息。介绍测试执行采用的测试工具，采用的测试级别(组件测试、集成测试、系统测试、验收测试)、采用的测试方法(白盒测试方法、黑盒测试方法、基于经验的测试方法等)，以及测试的主要类型(功能测试和非功能测试等)。同时描述还注意要包括以下几部分内容：软件配置、硬件配置、测试用例输入、操作步骤、输出、当时输出设备的相关输出信息和相关的日志等。其中软硬件环境的具体配置需要明确。比如：

(1) 软件配置说明包括操作系统的类型版本和补丁版本、当前被测试软件的版本和补丁版本、相关支撑软件，数据库软件的版本和补丁版本等。

(2) 硬件配置情况与被测试产品类型密切相关，需要根据测试执行的情况，准确翔实地记录硬件配置情况。比如影响软件产品测试结果的 CPU、内存和硬盘的相关参数，其他硬件参数根据测试用例的实际情况添加。如果测试中使用网络，还要记录网络的组网、容量、流量等情况。

2. 测试风险

测试执行总结报告中的测试风险部分介绍测试执行中的风险，以及测试计划的风险应对措施。这些信息为将来的测试执行实践提供参考，也可以用来更新测试团队的测试风险列表。比如以下测试风险信息：

1) 各个测试执行风险最终的状态分布。
2) 测试风险应对措施的及时性和有效性。
3) 遗漏的测试和缺陷风险的数量和影响。
4) 遗漏的测试和缺陷风险的原因分析。

3. 测试工作量

测试执行总结报告中还包含测试工作量收集和分析的结果，主要是测试活动的实际工作量和估算的工作量之间的比较和分析。估算的测试活动工作量可以直接从测试计划中得到，而实际的测试工作量可从每个测试人员的周工作汇报中获得。假如估算的测试工作量和实际的测试工作量之间差距较大，在测试执行总结报告中需要分析引起差距的原因，可能的原因如下。

1) 测试任务分解：在测试估算过程中，遗漏了重要的测试任务，或者对测试任务的复杂程度估计不足。

2) 测试估算依据：测试工作量在整个项目中的比例偏低，或者不符合当前软件项目对质量的要求。

3) 需求的变更：由于用户频繁的需求变更，导致开发团队不断地修改代码，测试人员不断地重复测试，从而增加了测试的工作量。

4) 软件的质量：由于在软件正式提交系统测试之前，没有很好地进行组件测试和集成测试，导致系统中存在太多的问题和缺陷，从而增加了测试的工作量。

5) 测试人员：由于测试人员对测试工具、测试过程、测试技术和测试对象等不了解，导致测试效率的低下，从而增加了测试的工作量。

对于每个测试阶段，收集实际耗费的测试工作量和估算的测试工作量是测试过程中重要的活动。测试工作量分析有助于将来项目测试计划的制定。假如可能，将耗费在评审、测试计划、测试设计、测试执行等活动中的工作量分开进行计算，这样就可以得到测试过程中不同测试活动的工作量分布。这些测试活动的工作量分布有助于将来测试过程的改进，以及修正已经定义的度量标准。

4. 测试执行

测试执行总结报告中的测试执行部分包括了计划的测试执行开始时间和结束时间、实际的开始时间和结束时间、计划的人力资源数目和实际的人力资源数目、计划的人力资源高峰数值和实际的人力资源高峰数值、计划的测试效率和实际的测试效率等。假如实际的测试执行和计划的测试执行差距比较大，在测试执行总结报告中需要分析引起差距的原因，可能的原因有：

1) 工作量估算：测试执行的工作量估算方面存在问题，由于估算的工作量和实际的工作量方面存在比较大的差距，从而导致测试执行的延误。

2) 测试风险：由于测试风险控制方面的缺失导致了测试执行进度的延误，例如：要求的测试仪表没有准时到达、第三方提供的软件没有按时提交等。

3) 需求变更：由于用户频繁的需求变更，导致开发团队不断修改代码，测试人员不断地重复测试，延误了测试执行。

4) 软件质量：由于在软件正式提交系统测试之前，没有很好地进行组件测试和集成测试，导致系统中存在太多的问题和缺陷，从而延误了测试执行。

5) 缺陷修复效率：由于开发团队对缺陷的重视程度不够，很多缺陷的修复都遗留到测试执行后期进行，导致确认测试和回归测试在较短时间内无法完成，引起测试执行的延期。

6) 测试人员技能：测试团队中的有些成员在技能方面无法达到组织的要求，导致测试执行进度缓慢。

对于测试活动来说，测试效率一般是基于测试用例数目进行测量的。而测试进度表一般是基于测试用例的数目和测试人员的测试效率来进行制定的。测试执行方面的偏差分析可以为将来的测试计划制定提供很好的输入，并且不断根据实际情况更新组织的测试执行能力，使得它们真实地反映测试人员的工作能力和效率。

在约定的测试执行完成后，测试经理需要总结此次测试执行的结果。表 7-7 是一个编写测试执行总结报告的简单模板和实例。

表 7-7　测试执行总结报告的实例

过 程 要 点	详 细 描 述
输入条件	1) 测试团队完成了预定周期的测试任务
工作内容	1) 测试经理根据此轮测试的结果，编写测试报告，主要应包含以下内容： 测试报告的版本 测试的人员和时间 测试所覆盖的缺陷——测试组在这轮测试中所有处理的缺陷，报告了测试经理处理的缺陷和实施工程师验证的缺陷。不仅要写出覆盖缺陷的总数，还要写明这些缺陷的去向 测试新发现的缺陷数量 上一版本活动缺陷的数量 2) 经过此轮测试，所有活动缺陷的数量及其状态分类 3) 测试评估——写明在这一版本中，哪些功能被实现了，哪些还没有实现，这里只需写明和上一版本不同之处即可 4) 亟待解决的问题——写明当前项目组中面临的最优先的问题，可以重复提出
退出标准	1) 在每轮测试结束之后应尽快将符合标准的测试报告发给全项目组
责任人	1) 测试经理

7.3.2　测试执行注意事项

在测试执行中需要注意以下几个问题：

1) 全方位的观察测试用例执行结果：　测试执行过程中，当测试的实际输出结果与测试用例中的预期输出结果一致的时候，是否可以认为测试用例执行成功了？答案是否定的，即便实际测试结果与测试的预期结果一致，也要查看软件产品的操作日志、系统运行日志和系统资源使用情况，来判断测试用例是否执行成功了。全方位观察软件产品的输出可以发现很多隐蔽的问题。比如：有时在测试嵌入式系统软件的时候，执行某测试用例后，测试用例的实际输出与预期输出完全一致，不过在查询 CPU 占用率时，发现 CPU 占用率高达 90%，后来经过分析，软件运行的时候启动了若干个 1ms 的定时器，消耗大量 CPU 资源，后来通过把定时器调整到 10ms，CPU 的占用率降为 7%。如果观察点单一，这个严重消耗资源的问题就无从发现了。

2) 加强测试过程记录：测试执行过程中，一定要加强测试过程记录。如果测试执行步骤与测试用例中描述的有差异，一定要记录下来，作为日后更新测试用例的依据；如果软件产品提供了日志功能，比如有软件运行日志、用户操作日志，一定在每个测试用例执行后记录相关的日志文件，作为测试过程记录，一旦以后发现问题，开发人员可以通过这些测试记录方便地定位问题。而不用测试人员重新搭建测试环境，为开发人员重现问题。

3) 及时确认发现的问题： 测试执行过程中，如果确认发现了软件的缺陷，那么可以毫不犹豫地提交问题报告单。如果发现了可疑问题，又无法定位是否为软件缺陷，那么一定要保留现场，然后知会相关开发人员到现场定位问题。如果开发人员在短时间内可以确认是否为软件缺陷，测试人员给予配合；如果开发人员定位问题需要花费很长时间，测试人员千万不要因此耽误自己宝贵的测试执行时间，可以让开发人员记录重新问题的测试环境配置，然后，回到自己的开发环境重现问题，继续定位问题。

4) 与开发人员良好的沟通：测试执行过程中，当你提交了问题报告，可能被开发人员无情驳回，拒绝修改。这时，只能对开发人员晓之以理，做到有理、有据、有说服力。首先，要定义软件缺陷的标准原则，这个原则应该是开发人员和测试人员都认可的，如果没有共同认可的原则，那么开发人员与测试人员对问题的争执就不可避免了。此外，测试人员打算说服开发人员之前，考虑是否能够先说服自己，在保证可以说服自己的前提下，再开始与开发人员交流。

5) 及时更新测试用例：测试执行过程中，应该注意及时更新测试用例。往往在测试执行过程中，才发现遗漏了一些测试用例，这时应该及时补充；往往也会发现有些测试用例在具体的执行过程中根本无法操作，这时候应该删除这部分用例；也会发现若干个冗余的测试用例完全可以由某一个测试用例替代，那么删除冗余的测试用例。

总之，测试执行的过程中及时地更新测试用例是很好的习惯。不要打算在测试执行结束后，统一更新测试用例，如果这样，往往会遗漏很多本应该更新的测试用例。

7.3.3 测试执行参考清单

不同的被测试软件产品或服务有各自不同的特性，因而测试执行的对象不同，测试执行的方式方法和侧重也应该有所不同。即使有很完善的测试用例体系和覆盖率，以及很高的自动化测试覆盖率，很多类型的测试也是要手动完成。比如在不同的浏览器下的用户界面测试。因此测试团队自己应该准备些测试执行常用的辅助工具，以提高测试执行效率和覆盖率。Checklist 就是清单。列出一条条要思考的具体内容。有很实用的价值。特别对于缺乏实际测试执行经验的测试执行人员特别有帮助。很多大的软件公司都有很多测试设计和执行时用的 Checklist。常常在公司测试团队内部网站或 Wiki 方式公布。以下举一个实例说明执行用户界面测试时 Checklist 的参考作用。在策划执行用户界面测试时，可考虑用表 7-8 所示的思考清单。

表 7-8　用户界面测试思考清单

序　号	思 考 清 单
1	应验证界面显示内容的完整性
2	应验证界面显示内容的一致性
3	应验证界面显示内容的准确性
4	应验证界面显示内容的友好性
5	应验证界面提示信息的指导性
6	应验证界面显示内容的合理性
7	界面测试时，应考虑用户使用的方便性
8	界面测试时，应考虑界面显示及处理的正确性
9	界面测试时，应考虑数据显示的规范性

表 7-9 可以用于指导用户界面测试具体执行的测试用例。

表 7-9　用户界面测试用例参考清单

序　号	参 考 清 单
1	菜单条是否显示在合适的语境中？
2	应用程序的菜单条是否显示系统相关的特性(如时钟显示)？
3	下拉式操作能正确工作吗？
4	菜单、调色板和工具条是否工作正确？
5	是否适当地列出了所有的菜单功能和下拉式子功能？
6	是否可以通过鼠标访问所有的菜单功能？
7	文本字体、大小和格式是否正确？
8	是否能够用其他文本命令激活每个菜单功能？
9	菜单功能是否随当前的窗口操作加亮或变灰？
10	菜单功能是否正确执行？
11	菜单功能的名字是否具有自解释性？
12	菜单项是否有帮助，是否语境相关？
13	在整个交互式语境中，是否可以识别鼠标操作？
14	如果要求多次单击鼠标，是否能够在语境中正确识别？
15	光标、处理指示器和识别指针是否随操作恰当地改变？
16	Tab 键的顺序，是否从上到下，从左到右(这种情况居多)？
17	逐一测试热键(快捷键)
18	Enter 键和 Esc 键的使用是否正常？
19	
20	

7.3.4　提高测试执行的水平

似乎执行测试的测试人员日常工作都差不多：都是按照测试计划，安装最新测试版本，执行测试用例，发现缺陷就报告，有问题就及时反馈给开发人员，验证修复的缺陷等。但事实上，即使在同样的测试环境和同样的测试用例，不同的测试人员执行测试，测试的结果总

会有差异。要提升测试执行的效率和质量，应注意以下十个方面。

1. 工作效率

举例来说，几个测试人员执行同样一组测试用例，都完成了测试，其结果为：员工 A 测试执行用了 3 天，测试出了 20 个 Bug；员工 B 测试执行用了 5 天，测试出了 50 个 Bug；员工 C 测试执行用了 3 天，测试出了 51 个 Bug。

对执行测试来说，输出结果的质量是决定性的因素。执行测试的人员要特别注意找到和应用最佳实践。思考怎样不断提高测试效率。不漏掉 Bug，又能快速完成任务。

2. 耐心

在执行测试用例过程中，有时需要在不同版本、测试环境反复执行同样的测试用例，这种工作是很枯燥的。需要有耐心，对用户负责，把好质量关。

3. 责任心

责任心是任何职业岗位都要求的职业素养。执行测试时往往没有人在旁边监督和检查执行情况。这就需要测试人员自觉自愿地对产品质量，对用户，对自己的工作负起责任。

4. 排查问题的能力

排查问题的能力主要指遇到异常和问题时，能分析情况，找原因，并找出最佳处理办法的能力。排查问题的能力依赖于对专业技术的理解能力和认真的态度。这依赖于经验积累。测试经验多的员工比新员工有优势，但是差不多时间进入团队的同事，对业务的熟悉程度不同在很大程度上取决于是否用心做事。团队成员间应该互相帮助，共同提高业务水平。

5. 回归测试的覆盖度

执行回归测试时应认真考虑和分析测试的覆盖度。通过了解开发人员怎样修复 Bug，分析 Bug 问题产生原因、修复影响的模块和功能以及其他相关因素。在测试用例库中选取回归测试用例，并执行回归测试用例。还要根据自动化测试覆盖程度决定手动测试的覆盖程度。

6. 敏捷测试模式的效率

执行敏捷测试是最考验测试工程师的一种测试任务。常规的测试，我们可以依赖于完整的测试策略和测试计划、规格学习和讨论、测试用例编写评审、测试执行、Bug 分析和各种控制方法。但是敏捷开发项目中的测试却常缺乏需要的时间、文档和其他资源。比如前端的交付件可能不够全面，测试策略和测试用例也可能来不及构建。所以要求测试工程师熟悉系统框架，熟练应用各种测试工具。

7. 注意细节

注意细节是一个比较务虚的词，没有特别具体的量化指标来考核。主要是要特别细心。注意细微的现象和功能行为。这意味着对被测软件功能和系统的敏感度，对细节的敏感度。

比如图像质量的测试，彩色的图像忽然变成黑白的，可能任何一个测试人员都能发现问题。但是每隔 30 秒，图像忽然颤抖一下，则只有注意细节和很细心的人才能捕捉到这种细微

的异常。

8. 提高自动化测试覆盖度

在可能的条件下，应尽量不断增加测试的自动化覆盖率。因此在执行测试用例时应积极思考怎样实现手动测试用例的自动化。

9. 不断自我提高

执行测试的过程是可以不断思考怎样做得更好、更快、更准确。测试的技术水平也会因为不断的思考改进而得到很多的进步。

10. 提高业务熟练度

对被测软件业务知识的掌握和理解程度在产品线的测试团队中，是根本也是核心。要保证测试执行具有预期的效果，在有限的时间和资源条件下，熟知产品功能行为的测试人员才能保证做好执行测试的任务。

除了了解测试计划和测试规格外，测试执行人员还需要掌握必备知识和技能。不同的测试对象，需要学习不同的知识和技能。主要包括：

1) 领域知识：不同的应用软件需要不同的领域知识(通信、电子商务、金融证券、物流等)，必须对被测系统的领域有一定了解，掌握相关知识。领域知识的掌握可以帮助测试人员理解和运用软件产品的工作原理和工作过程，更好地执行测试，使测试环境的搭建更加符合实际的使用环境等。

2) 测试知识：测试知识包括基本的测试过程、测试技术、测试理念等，例如：对于白盒测试方法，测试人员必须掌握各种白盒技术的基本理论。

3) 网络知识：若测试对象是在网络环境下执行，则必须对所要求的网络知识有所了解。

4) 系统知识：若测试对象在特定的系统中运行(如特定硬件、特定数据库、特定接口系统等)，则必须了解相应的系统知识。

习题与思考题

1. 软件测试执行的内容包括哪些？
2. 影响测试执行的因素有哪些？
3. 怎样判断测试执行的结果是通过还是失败？
4. 测试中使用出口准则的目的是什么？出口准则可以使用哪些度量进行判断？
5. 请解释下测试过程中的四个度量指标？
6. 测试执行控制阶段的主要测试活动有哪些？
7. 根据你的理解说明怎样提高测试人员的测试执行能力？

第 II 部分　HP ALM 工具入门篇

本部分是 HP ALM 的入门基础。HP ALM 是软件测试专业基础知识的具体应用，是应用型软件测试人才培养的基本要求，也是软件测试的一项基本技能，要求全面掌握和深刻理解。为此，需要投入一定的学时来学习理论和实践知识。

第8章　HP ALM介绍

8.1　HP ALM 概述

惠普应用程序生命周期管理(HP Application Lifecycle Management，以下简称 HP ALM)系统是一个复杂的过程管理系统。无论你的组织架构是敏捷、迭代还是瀑布，有效的生命周期管理可以使得应用程序具有更好的可预测性、更高的重复性、更好的质量和更强的应变性。理解项目里程碑、交付产品、资源和预算需求，并持续追踪项目的健康、标准和质量指标，有助于交付经理实现这些目标。

HP ALM 系统使你能管理应用程序核心的生命周期，从需求开始贯穿整个开发过程。HP ALM 帮助你在资料库中心为应用程序生命周期管理工作流创建框架和基础，同时支持分布团队之间的沟通与合作。

ALM 有以下特点：

1) 为所有测试个体提供基于 Web 的知识库，并为整个测试流程提供清晰的指导；

2) 在应用程序生命周期的每个阶段之间建立无缝集成和顺畅的信息流；

3) 支持对测试数据和覆盖范围的统计分析，提供应用程序生命周期每个时间点的进度和质量图。

8.1.1　ALM 版本

ALM 提供如表 8-1 所示的版本。

表 8-1　ALM 版本

ALM 版本	功　　能
HP ALM	HP ALM 完整功能，为成熟的组织和 COE 的企业发布管理提供应用程序生命周期和评估质量的核心功能
HP Quality Center 初级版	为应用团队管理最多有五个并发用户的小型发布提供的版本
HP Quality Center 企业版	为质量管理团队管理中大型发布提供的版本
HP ALM 性能中心版	为管理大规模性能测试项目的所有方面提供的版本

其他版本和完整版的主要区别是完整版提供了项目之间的共享特征。具体特点包括导入、同步和共享库文件，以及共享缺陷、实现跨项目定制。本课程以 HP ALM 11.5 版本为例。

8.1.2　ALM 流程

如图 8-1 所示，ALM 的应用程序生命周期管理流程分为以下几个阶段：

1) 指定版本：制定一个发布周期管理计划，更高效地管理应用程序发布和周期。追踪应用程序发布，并根据计划确认发布是否正常。

2) 指定需求：分析应用程序并确定需求。可以跨多个发布和周期管理需求，并在需求、测试、缺陷之间实现多维追踪。ALM 为需求覆盖、关联到质量评估和商业风险中的缺陷提供实时可见的功能。

3) 计划测试：创建一个基于需求的测试计划，ALM 为手动和自动测试都提供了知识库。

4) 执行测试：创建测试集，完成测试运行。ALM 支持健壮测试、功能测试、回归测试和其他测试。根据计划来执行测试，从而识别和解决问题。

5) 追踪缺陷：报告在应用程序中检测到的缺陷，跟踪修复进程。分析缺陷和缺陷趋势，帮助做出合理的“执行/不执行”决策。ALM 支持完整的缺陷生命周期——从初始问题检测到修复缺陷以及确认缺陷修复。

每个阶段都可通过生成详细报告和图表来分析数据。ALM 是一个基于 Web 的工具，可给基于 Web 的项目创建知识库。它是一个 Java EE 开发的 C/S 组织结构，是一个服务的集合体。

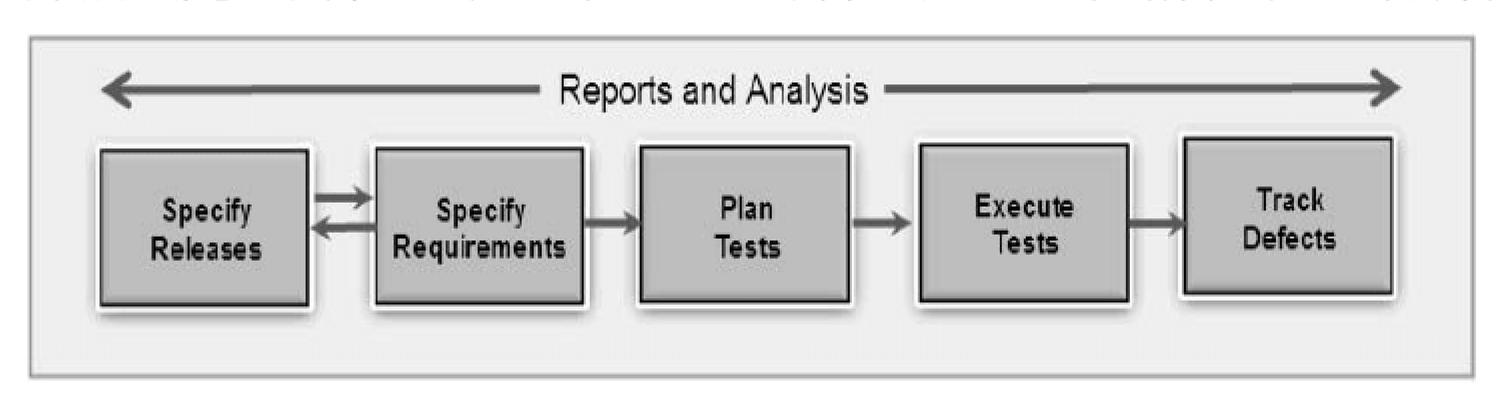


图 8-1　应用程序生命周期管理流程图

一个端到端的实例

假设某部门的软件开发团队已经开发了 Flight Reservation 程序的 Release 4.0 版本。这个系统提供用户下订单、更新或删除订单、搜索航班的功能。如图 8-2 所示。

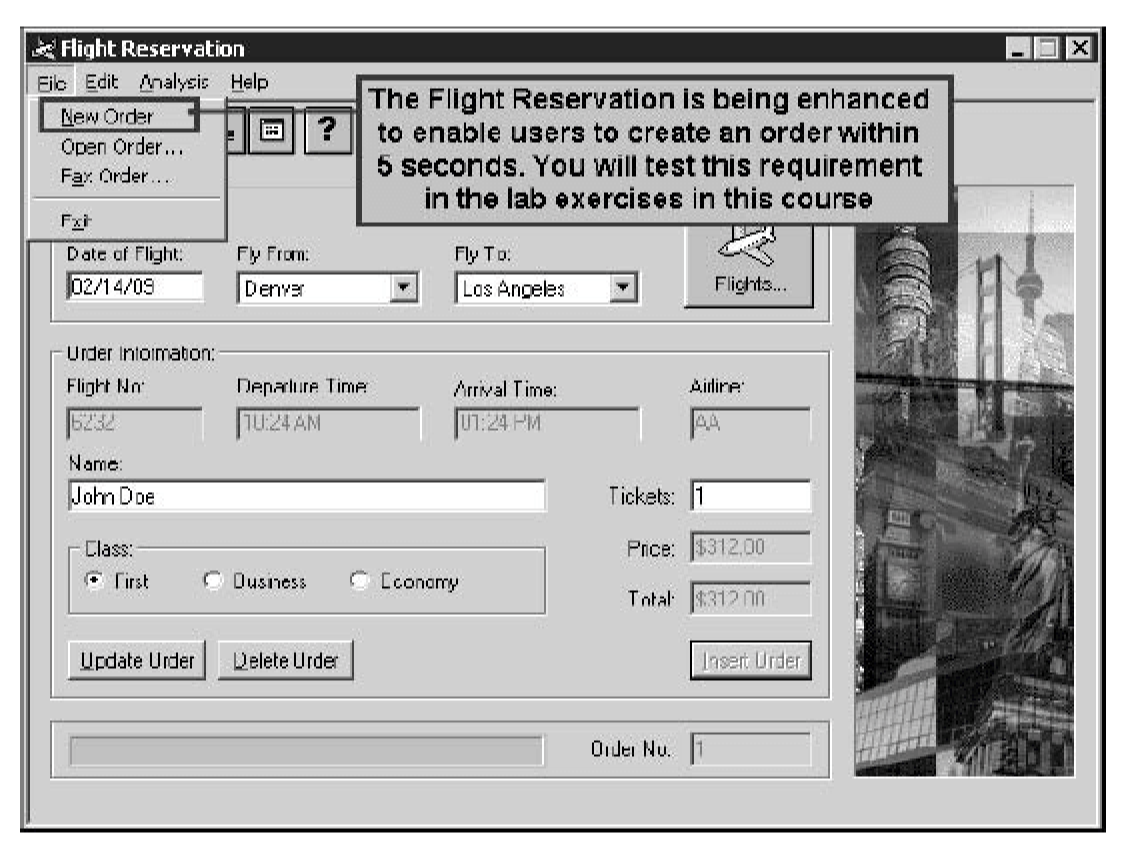


图 8-2　创建业务流程的顺序(订单)

现在不仅需要测试 Release 4.0 的新特性，也要测试现有的业务流程功能是否和以前保持一致(回归测试)。部门的质量经理要求 Release 4.0 在 30 天之内可以使用，那么你将需要进行 4 个测试循环。

公司以前一直用 Excel 跟踪业务需求，但是管理人员决定转移到 ALM，因为 ALM 可以管理所有类型的需求，而不仅是测试需求。首先你需要定义需求，然后根据业务的关键程度和需求的失效概率来分析项目风险。

然后，你需要创建手动或自动的测试用例来测试需求。因为测试会直接安排到需求中，然后可以跟踪需求的覆盖率，执行测试，关联没有覆盖到的需求和测试实例的缺陷。

最后，可以生成图表和报告来分析项目状态。

为测试 Release 4.0 版本，你需要：

1) 创建一个发布树，包括测试新特性的测试周期，已经存在的 Flight Reservation 系统的功能和性能。

2) 创建详细说明功能和性能标准的需求。

3) 在需求之间增加追踪链接，保证一个需求的任何变更都可以及时对相关需求进行更新。增加追踪链接后，分析与需求相关的风险，确定需求应该、必须或者不必测试。

4) 定义一个分层次的树形库来创建和管理库。

5) 创建一个基于你所定义的功能和性能需求的测试计划。

6) 执行测试以确认需求是否满足要求。

7) 为失败的测试步骤记录缺陷。

8) 生成报告和图表来查看需求和测试计划数据。

登录 ALM 的过程，首先是用户授权登录，然后分配访问域和项目的权限：

1) 从程序菜单中选择应用程序并打开，或者双击桌面上的快捷方式。出现应用程序生命周期管理的主界面。如图 8-3 所示。



图 8-3　登录到 ALM

2) 在主页单击 Application Lifecycle Management 链接，出现登录界面。

3) 在 Login Name 和 Password，分别输入相应的用户名和密码。

4) 单击 Authenticate，在 Domain 和 Project 列表中，分别选择域和工程。

5) 单击 Login 按钮。

8.2　ALM 模块

1) Dashboard：包含以下模块(如图 8-4 所示)：

(1) Analysis View：创建图表，报告和 Excel 报告。

(2) Dashboard View：创建 Dashboard 页面，以便在单个显示页面中查看多个图表。

2) Management：包含以下模块：

(1) Releases：为应用程序管理流程定义发布和周期。

(2) Libraries：定义库，追踪项目中的变化、重用项目实体或跨多个项目共享实体。

3) Requirements：指定测试需求。

4) Testing：包含以下模块：

(1) Test Plan：基于 Requirements 模块中定义的测试需求来创建测试计划。

(2) Test Resources：管理测试中使用的资源。

(3) Test Lab：在应用程序中运行测试，分析结果。

5) Defect：缺陷模块，可报告应用程序中的设计缺点，并在应用程序管理流程的各个阶段跟踪源于缺陷记录的数据。直到应用程序开发者和测试人员确定已解决缺陷。

根据 ALM 的许可号和设置，你可能可以进入以下模块：

1) Business Models：在 Requirements 菜单的下面，从其他应用程序中导入业务流程模型，并测试这些模型和它的组件的质量。

2) Business Components：在 Testing 菜单下，用来设计合并业务组件到业务流程测试中。

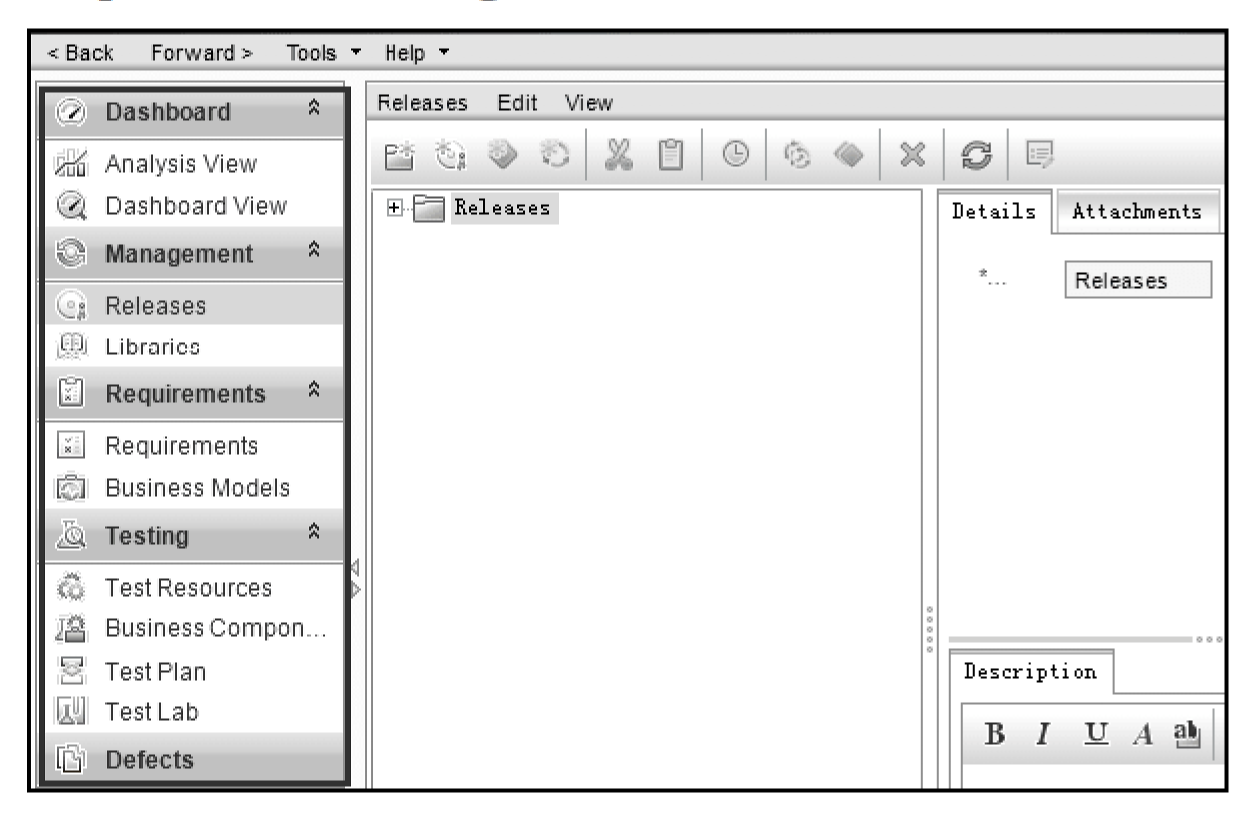


图 8-4　ALM 模块

8.3　ALM 工具栏

当登录到 ALM 时，项目主页会显示你上次用过的模块。项目窗口提供下列工具(如图 8-5 所示)：

1) Common Toolbar：提供导航按钮，常用工具的访问，文档和附加资源。这个工具在窗口的右上角。

2) Module Toolbar：包括在当前 ALM 模块中常用的命令按钮。

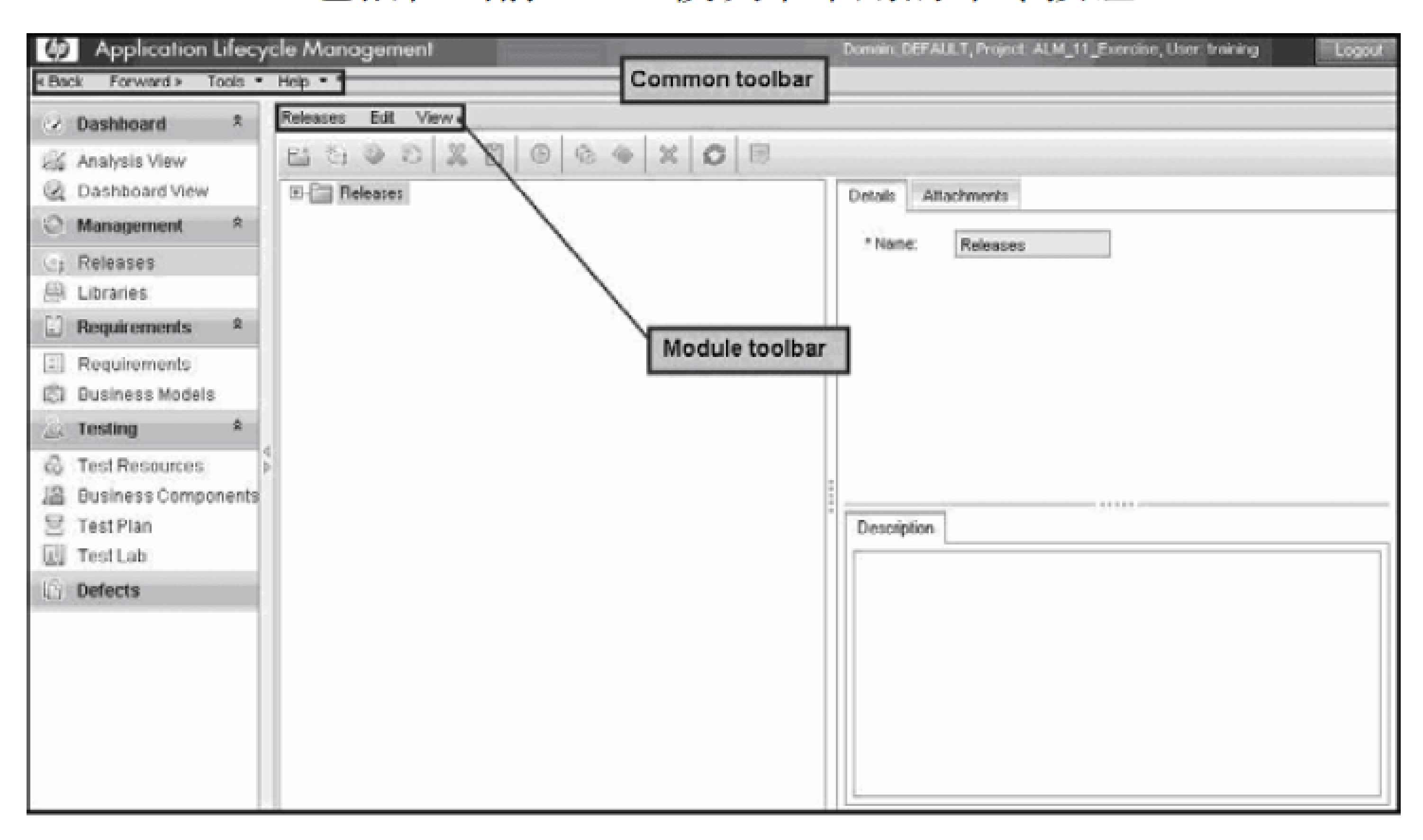


图 8-5　ALM 工具栏

8.4　快捷菜单

在 ALM 的界面右击某个界面组件的时候会显示相关内容菜单。这些菜单都是所选的界面组件的快捷命令菜单。见图 8-6。

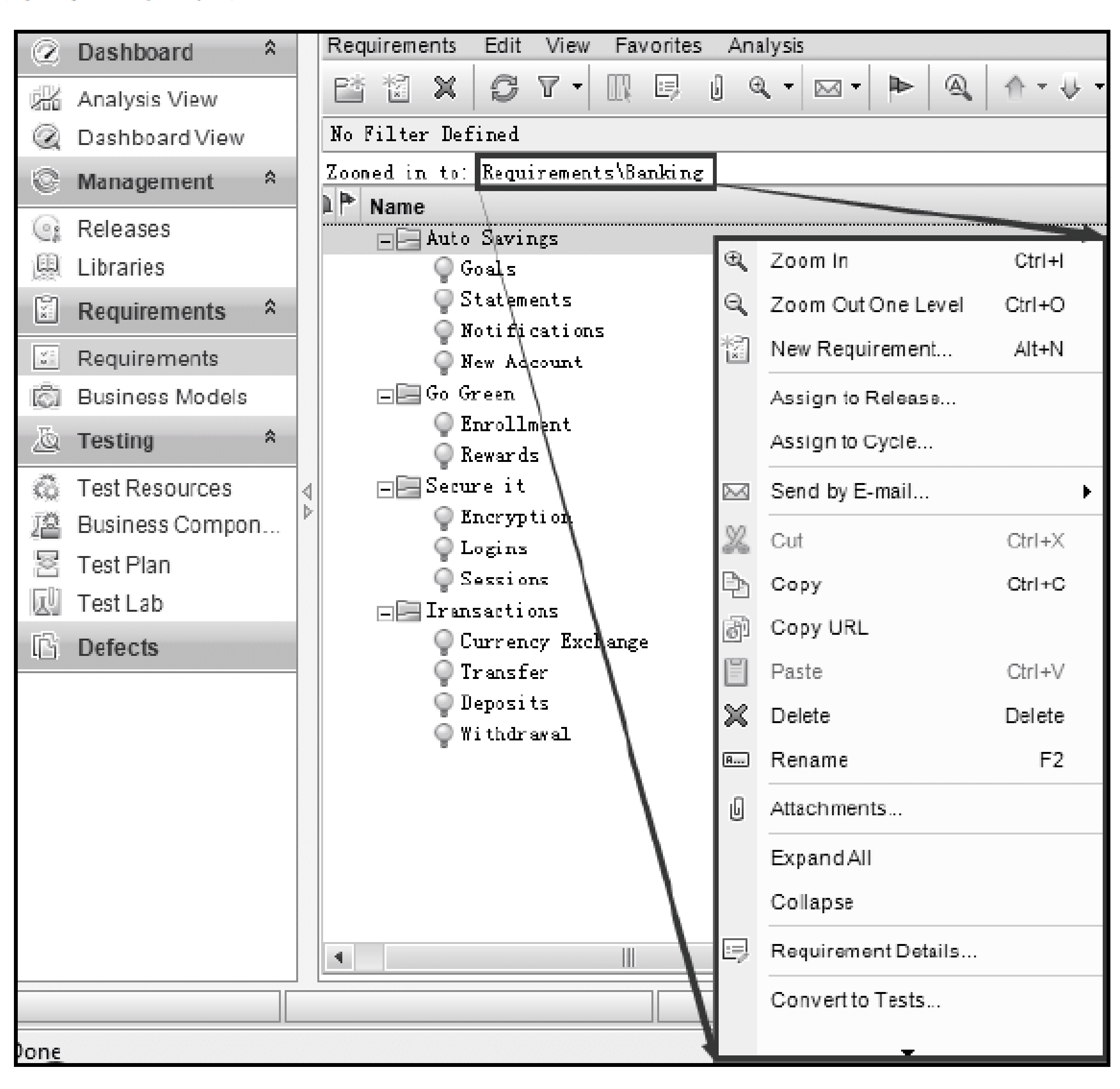


图 8-6　需求模块的快捷菜单

8.5　工具和帮助

如图 8-7 所示，ALM 的常用工具栏提供以下菜单：

1) Tools：

(1) 不用退出 ALM 直接切换到另一个项目。

(2) 定制项目。

(3) 记录缺陷。

(4) 核查模块拼写并配置拼写核查选项。

(5) 清除项目历史。

(6) 生成 Word 和 Excel 报告。

2) Help：打开 ALM Documentation Library 和其他在线资源，并显示 ALM 版本。

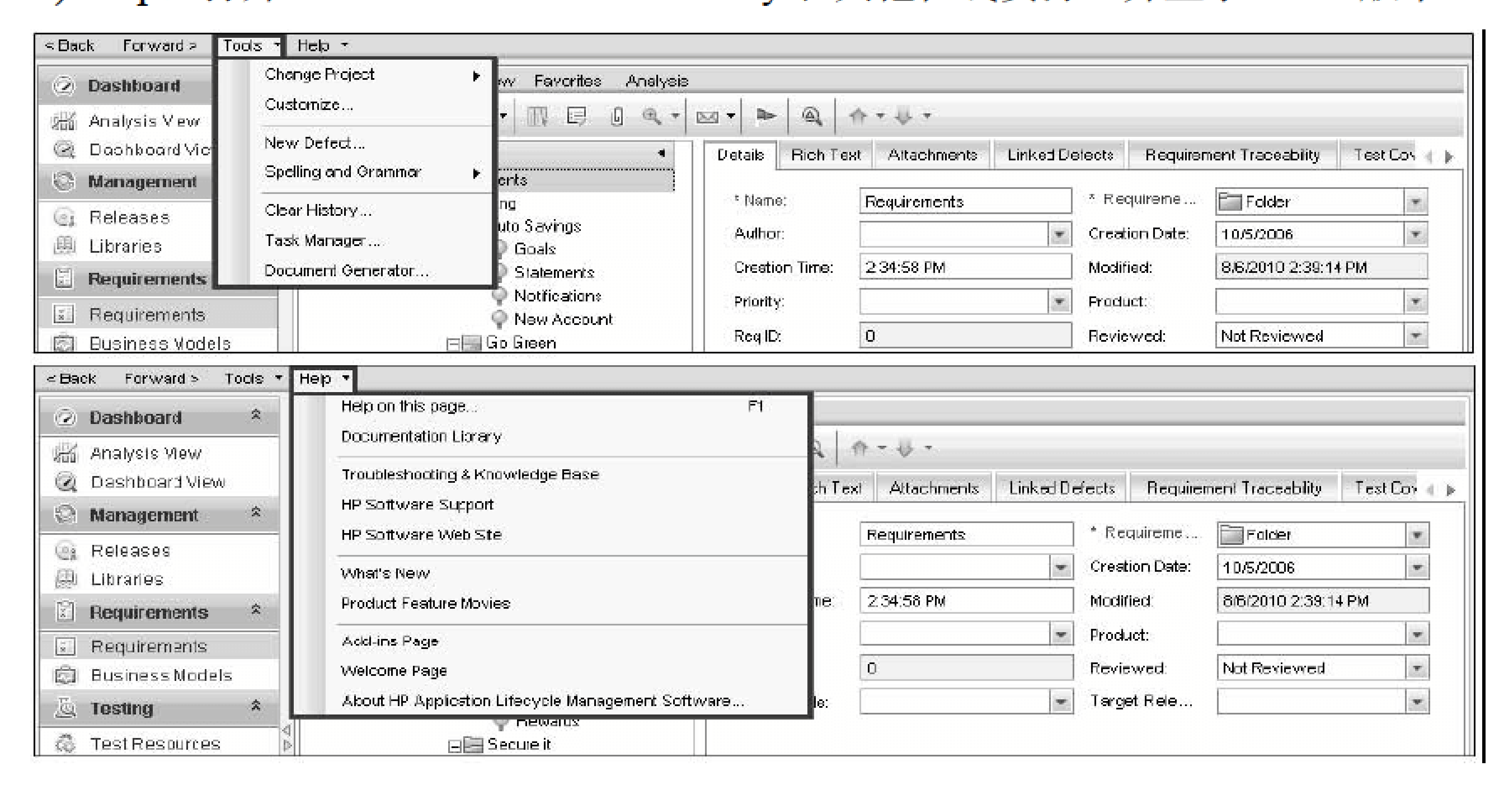


图 8-7　工具和帮助菜单

8.6　使用收藏视图

你可以在当前视图中保存所有设置为收藏视图，包括设置过滤的条件。如图 8-8 所示。

1) 保存收藏视图：从任何模块的菜单工具中，选择 Favorites | Add Favorites，显示 Add Favorite 对话框。

2) 在 Add Favorite 对话框的 Name 字段中，为视图输入一个名字。

3) 指定视图是私有还是公有。私有视图只有创建该试图的用户才可见，公有视图对 ALM 的所有用户都可见。同时，使用私有视图是查看只有你使用的数据很好的方法，提高了数据的安全性。另一方面，对于有多个用户使用的数据，公有视图是一个更好的选择。这样保证了数据的一致性，因为所有的用户看到的是相同的数据字段。

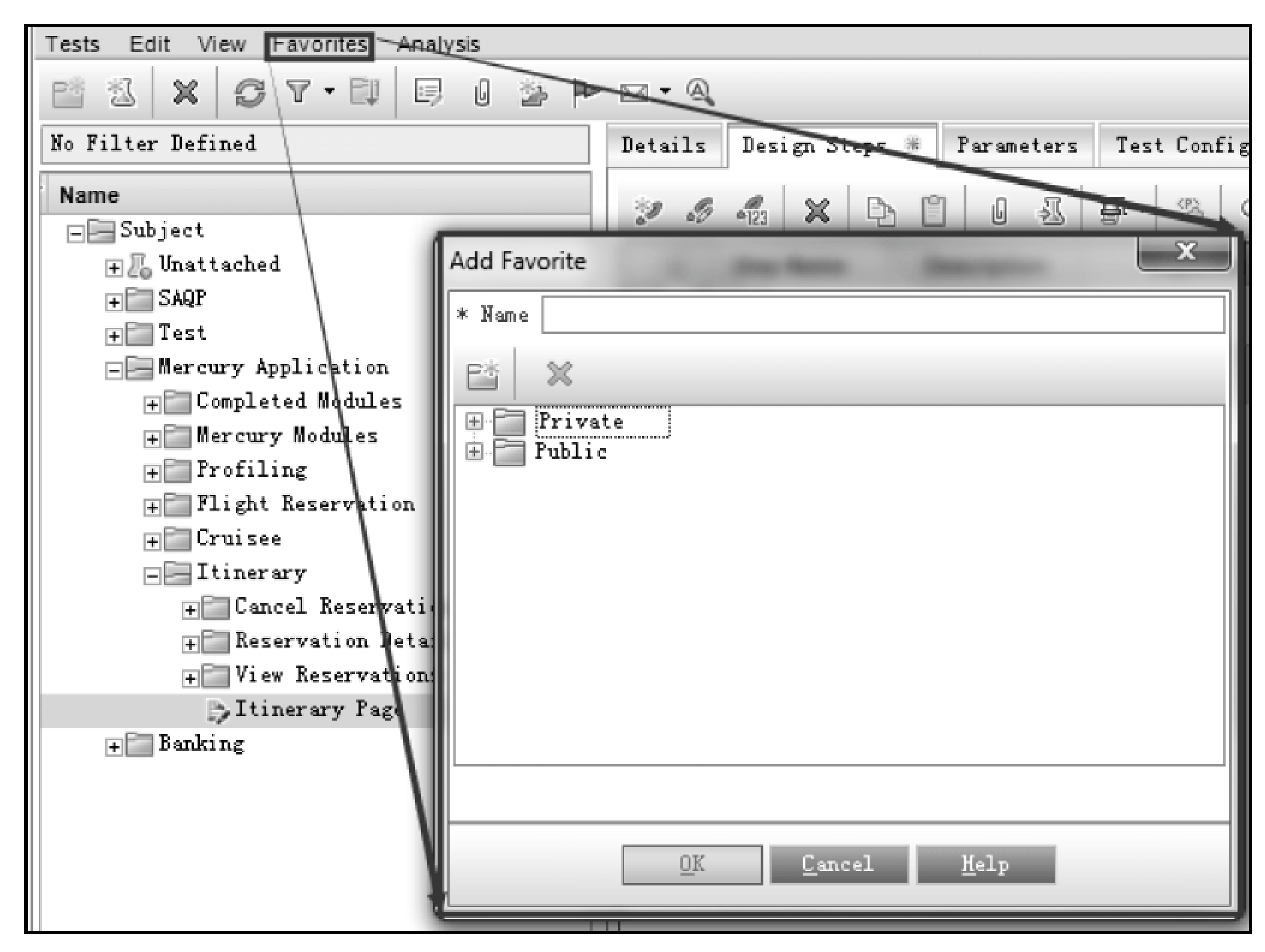


图 8-8　Favorites 视图

此外还要注意登录功能：

当有一段时间没有操作 ALM 的时候，登录会过期。以前，这需要你退出然后再次登录。登录之后重载定制的项目会花费一定的时间。现在，当你的登录到期，会被提示重新连接。如果项目管理员没有对项目做任何重要的改变，客户化定制就不会重载，使你可以快速重新连接并继续上次离开时候的工作。

习题与思考题

1. ALM 包括哪些常用模块？
2. ALM 常用工具栏在哪里？它支持哪些功能？
3. 怎样看到新特征在一个发布中的各个流程？
4. 从 ALM 中如何进入用户帮助站点？

练习：分析被测的应用程序

为在 ALM 中正确地创建项目，必须理解业务流程，这是应用程序的一部分。对这些流程的良好理解可以更好地利用 ALM 的特性。

第9章　发布工作

9.1　发布概述

惠普应用程序生命周期管理(ALM)通过定义发布和周期来组织和跟踪即将进行的发布。测试流程始于在 Management 模块中定义 Releases，如图 9-1、表 9-1 所示。该模块用于根据项目需求、测试和缺陷，排列业务的优先级别和质量期望。

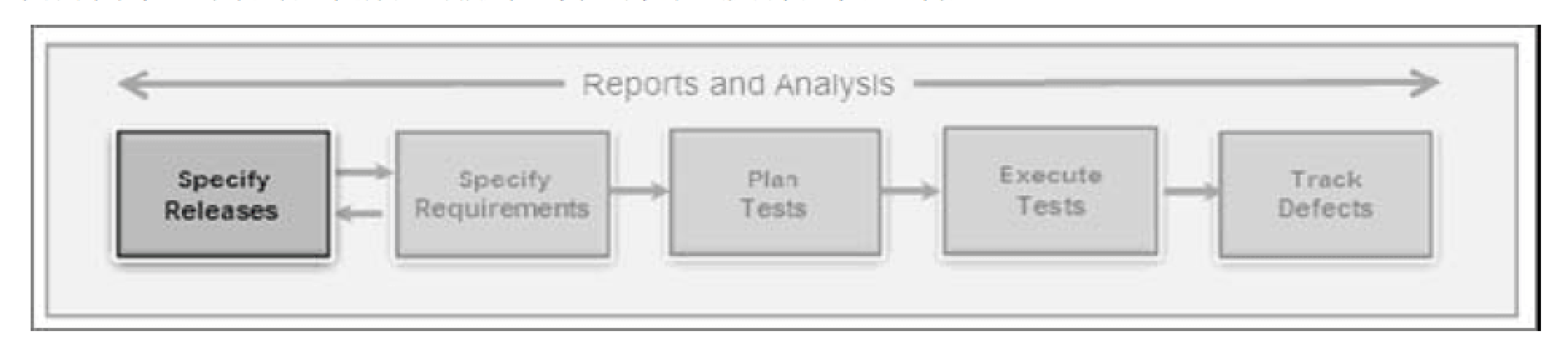


图 9-1　指定发布

表 9-1　Release Module 的用户界面元素

用户界面元素(A-Z)	具 体 描 述
<Releases 模块通用用户界面元素>	Release 字段：字段定义。Release 菜单和按钮：指令和按钮描述。 Release 图标：图标描述。 ALM 主菜单和侧边栏：细致的工具菜单、帮助菜单、侧边栏
< 发布树>	发布的图形表示
Attachments 标签	列出包含当前选中的文件夹、发布、或周期额外信息的附件
Description 标签	描述当前选中的文件夹、发布或周期。单击文本框，显示格式工具栏和文本拼写检查
Details 标签	显示当前选中的文件夹、发布、或周期的细节。在 Statistics 区域显示与发布和周期相关的需求和测试集文件夹
Master Plan 标签	使用甘特图显示发布的进程。 ALM Editions：对于 Quality Center 的初始版、Quality Center 企业版、Performance Center 版是不可用的

业务部门(LOB)是组织内的独立业务单元，它有自己成套的规则、标准、资源、目标和应用。例如：图 9-2 对 Online Banking 和 Credit Card 这两个银行里面同步开发的软件业务部

门进行了说明。

现行的 Online Banking (v10.0) 版本使用户能够进行基本的网上银行交易，如查看支票、储蓄、信用卡账户和清单，在储蓄、支票账户和工资单之间转账。Online Banking 的 10.5 版本将提供新的功能，如在线储蓄，还有共享交易账户。

在 ALM 里，一个应用程序的版本被称为发布，它表示应用程序在同一时间作出的有益于用户的一组变更。当 Online Banking 和 Credit Cards 两个部门分别开发其软件的不同版本的时候，发布协调两个业务之间的测试活动。

一个版本的发布是要在指定的时间内开发完成的。例如：考虑 Online Banking 应用程序的 10.5 版本必须在 6 个月内被开发出来。在这 6 个月内，发布贯穿一系列的测试周期。在 ALM 里，一个测试周期被称为一个周期。一个周期是一组开发人员和质量保证人员在发布的时间范围内的所有用来实现共同目标的时间和精力代价。

发布和周期均定义了起止日期。每个周期都有一个特定的目标。例如：考虑 Online Banking 应用程序的 10.5 版本贯穿四个周期。第一个周期测试 10.5 版本包含的新特征。第一个周期结束后，已确保新特征包含在新版本中而且能正常工作。

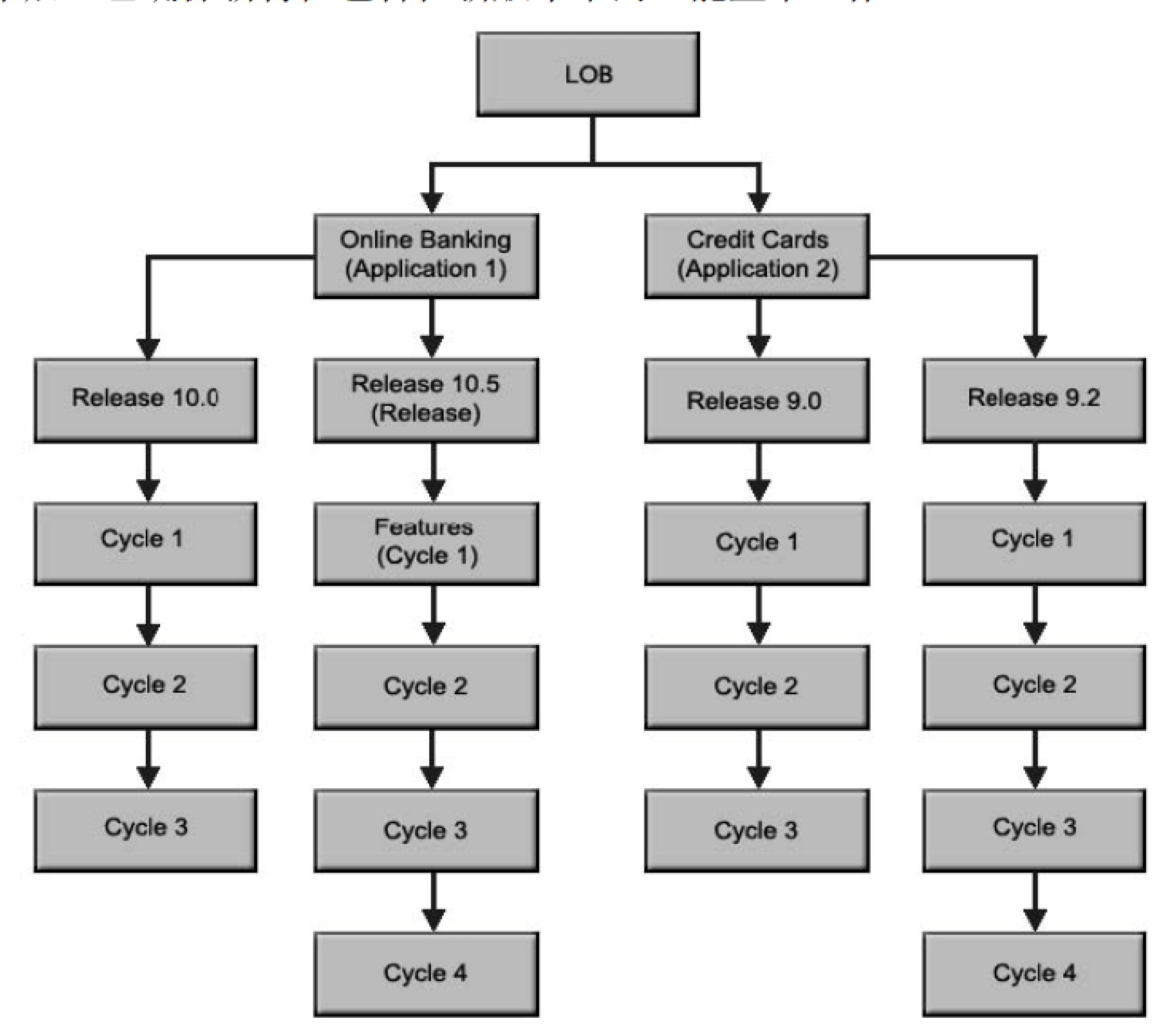


图 9-2　LOB、Relase 和 Cycle 的关系

第一个周期结束后，发布进行后续测试周期来测试应用程序的功能和性能等内容。开发团队修复测试团队在周期内记录的缺陷。缺陷修复后，测试团队依赖于组织里运用的测试流程，确认缺陷修复并在本周期或后续周期中关闭缺陷。不同应用程序的周期数不同，这依赖于发布变更的数量。在所有周期都完成后，着手准备把版本发布给客户。

通过在 Management 模块中定义发布从而开始测试进程。 Management 模块的基础是发布树。按照层次结构使用发布树，规范程序中要加入的发布。发布树中包含一个发布文件夹。

发布文件夹中包含程序的不同发布。发布由周期组成，每个周期都有一个预定义的目标。发布的起止日期必须包含发布中的所有周期。但是，周期之间可以是独立存在的，如果它们之间没有干扰，也可以重叠。举例来说，项目经理可以决定使回归测试周期和性能测试周期重叠。

例如，图 9-3 显示了 Flight Reservation 程序的发布。显示 Flight Application 文件夹的树包含版本 4.0 和它的周期：

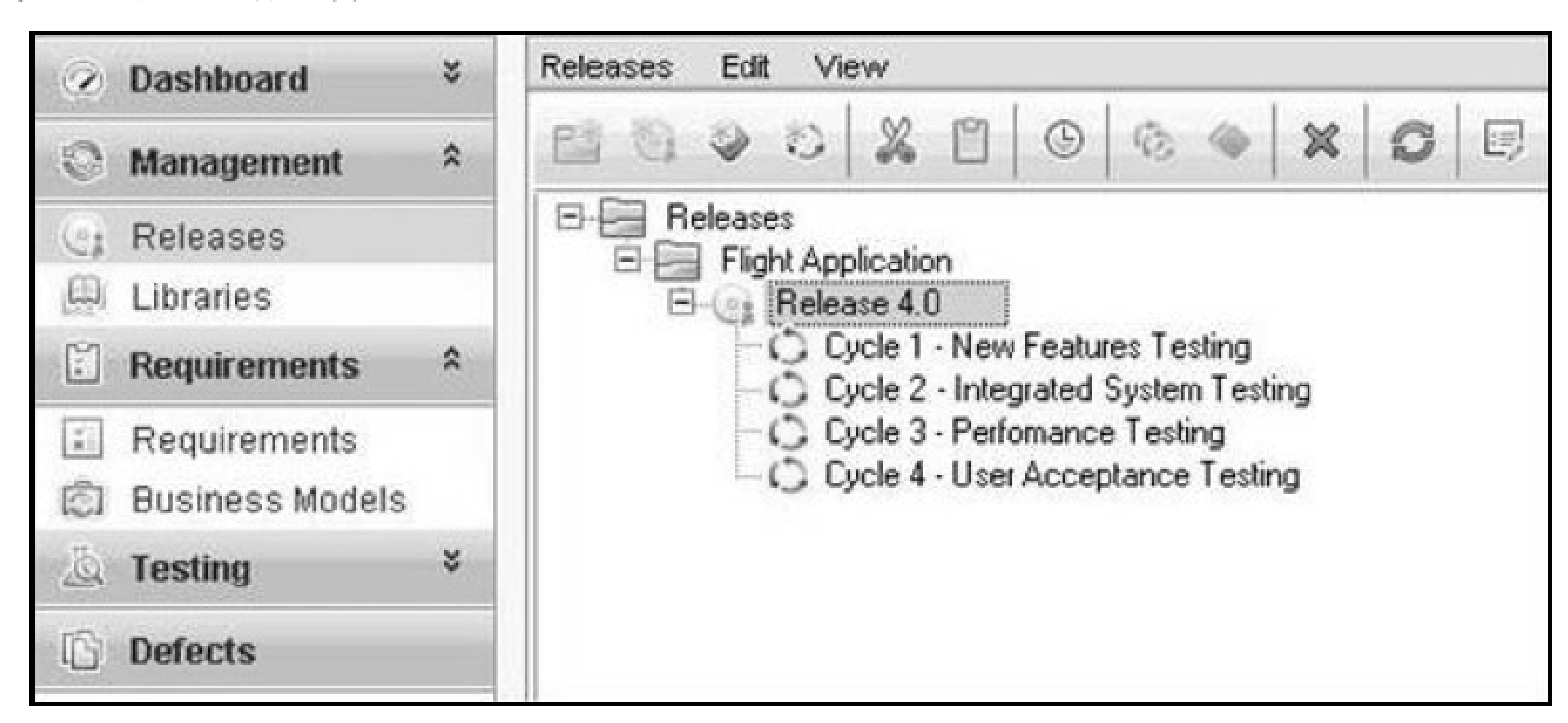


图 9-3　发布树示例

1) Cycle 1-New Features Testing：测试此次发布的程序中实现的新特点。新特点测试后，开发团队修正测试中记录的缺陷。但是，根据缺陷的危急程度和严重性，一些缺陷可能会等到下一个周期才修正。

2) Cycle 2- Integrated System Testing：这个周期中测试新功能可能对原有功能产生的影响。

3) Cycle 3-Perfomance Testing：根据一些标准来测试应用程序的性能，例如程序支持的并发用户数量和程序响应时间。

4) Cycle 4-User Acceptance Testing：这一周期确保新产品满足业务期望。

9.2　创建发布

通过创建发布树来定义发布的分层架构。根据组织架构的测试流程，可以为每个程序创建一个发布文件夹，或多个程序创建一个文件夹。发布树工具可参见表 9-2。

默认情况下，Management 模块的左面板显示 Releases 文件夹。这是个预定义文件夹，不能删除。如果需要，可以根据需求重命名 Releases 文件夹并向其中添加发布。创建新文件夹时，就在 Releases 文件夹中创建了一个新的级别。

在 New Release 对话框，可定义一个新的发布。在 Release Details 对话框可以查看和更新所选发布的详细内容。见图 9-4。

表 9-2　创建发布树的工具

图　标	按 钮 名 称	功　能
	新建发布文件夹	在发布树中创建发布文件夹
	新建发布	在发布文件夹中创建发布
	新建周期	在发布中创建周期

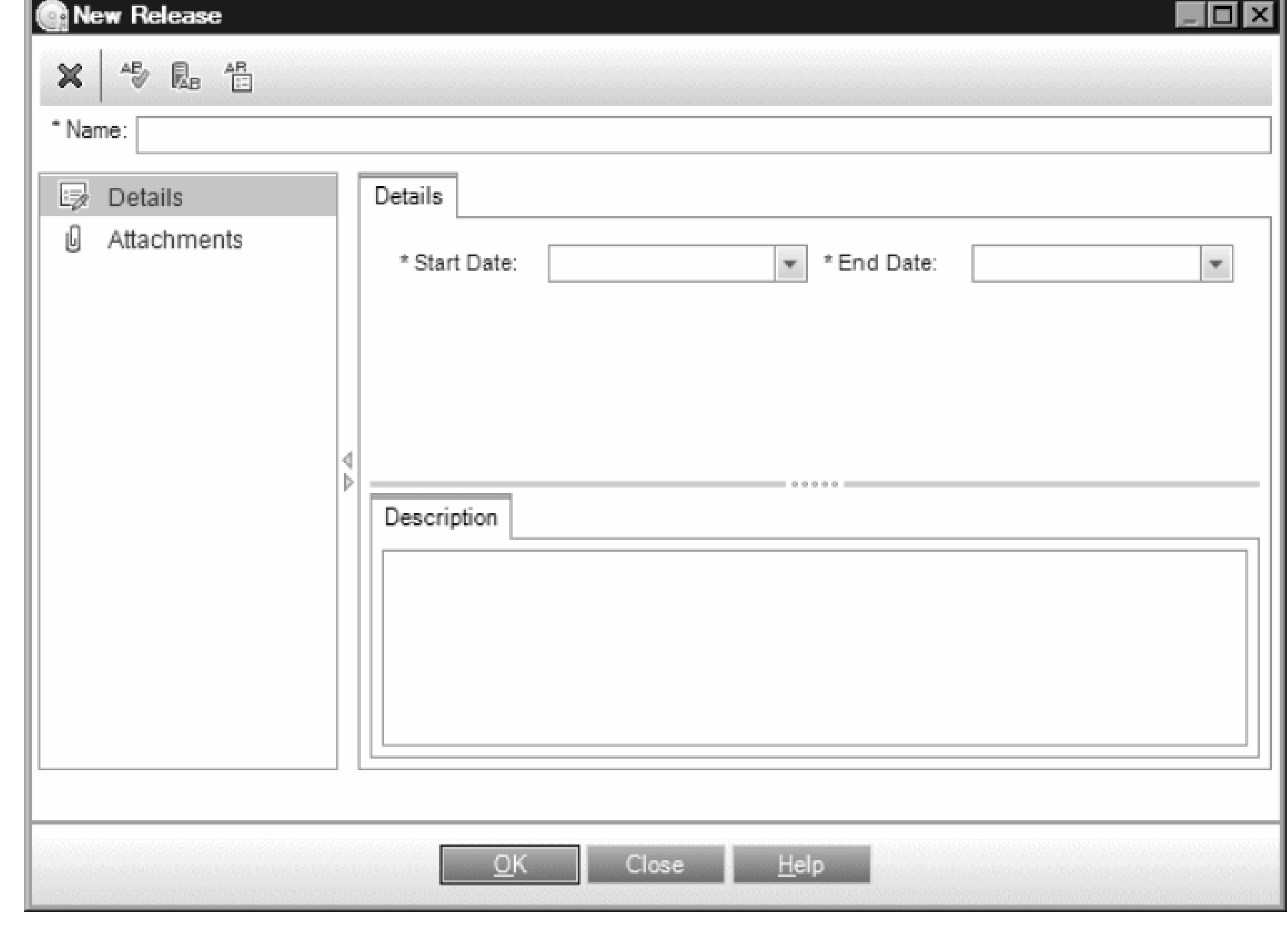


图 9-4　新建发布

9.3　创建周期

向发布添加周期后，使用 Details 标签指定周期详细信息。详细信息包括开始日期，结束日期和周期的简短描述。指定周期的详细信息：从发布树中，选择要指定细节的周期。选中周期的 Details 页出现在 Management 模块的右边面板中。如图 9-5 所示。

在 Details 页面，执行下列任务：

1) 从 Start Date 列表中选择周期开始日期；

2) 从 End Date 列表中选择周期结束日期；

3) 在 Description 字段，输入周期期望完成的目标的描述。

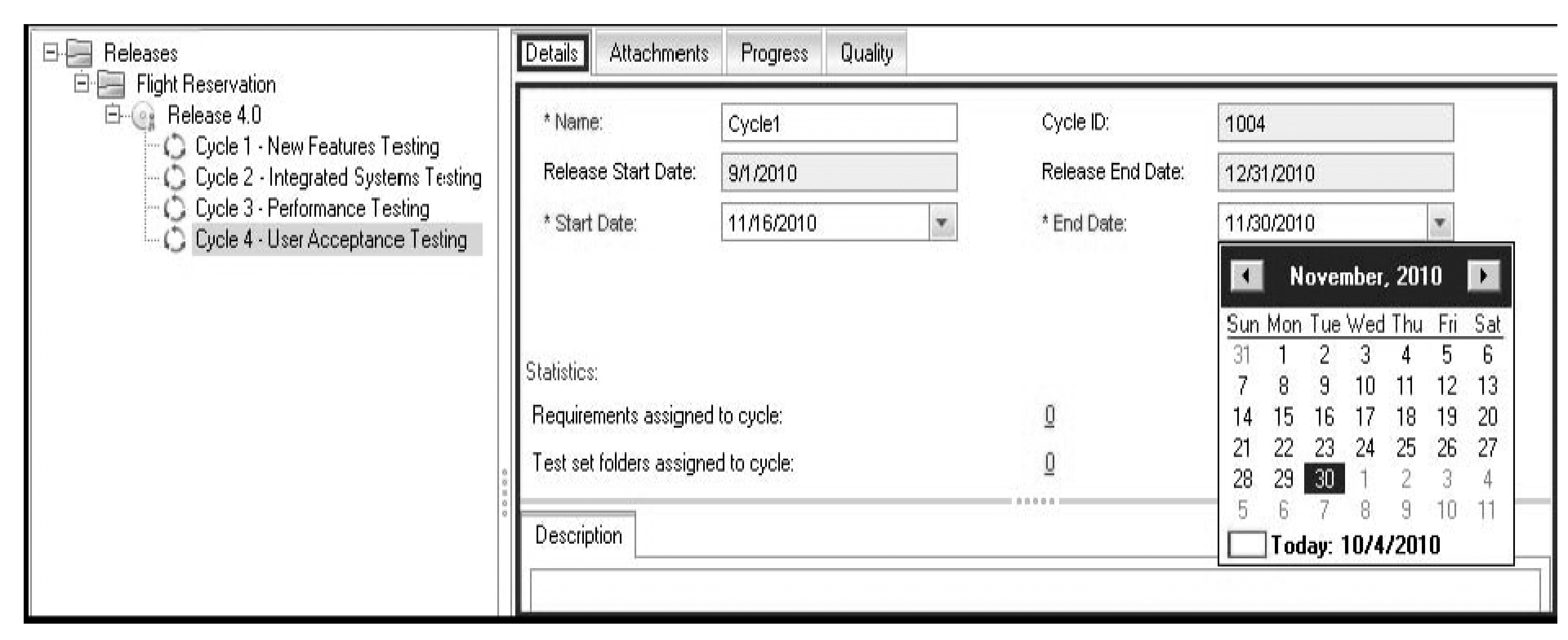


图 9-5　指定周期的细节

9.4　重新计划发布/周期

Reschedule Release 对话框可用来重新安排发布、周期的开始和结束日期。如图 9-6 所示。

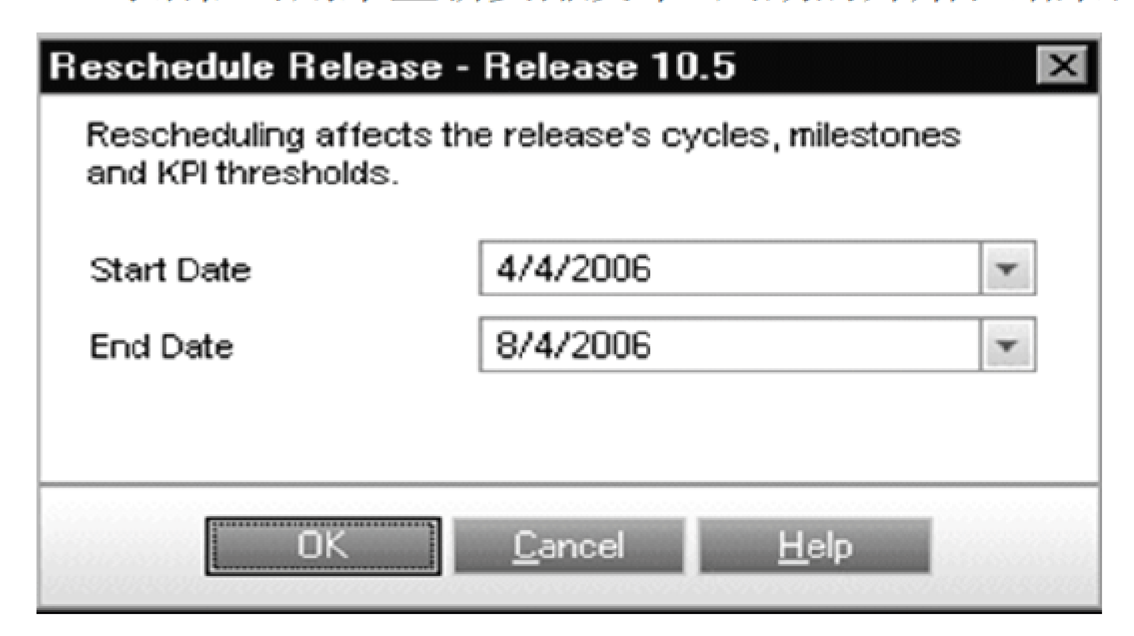


图 9-6　重建计划发布

9.5　添加附件到周期

向周期添加附件时可能会有环境要求。例如，使用 Microsoft Excel 安排发布的时间和资源评估，并且想将这个电子表单附加到发布的周期中，使开发者能够跟踪时间。要使用 Attchment 标签向周期添加附件。如图 9-7 所示。

向周期添加附件：

1) 在 Management 模块的右面板，单击 Attachments 标签；

2) 在 Attachments 页的工具栏中，单击关联需要的附件类型按钮。

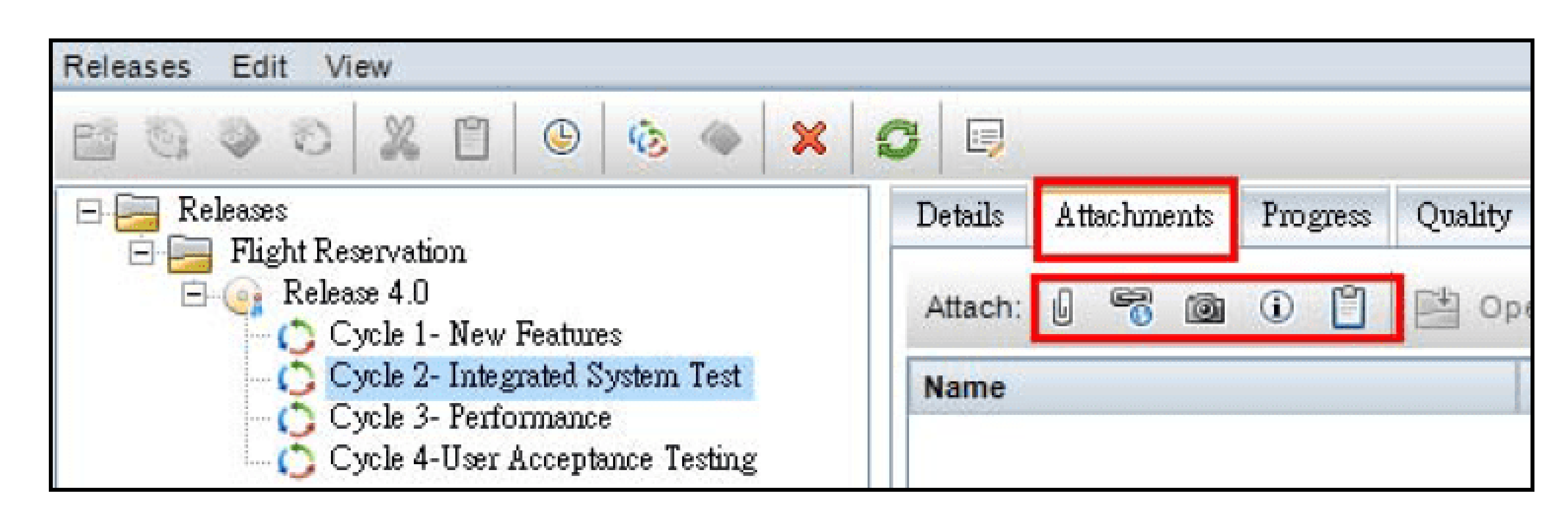


图 9-7　向周期中添加附件

附件类型请参见表 9-3。

表 9-3　附件类型

图　标	名　称	用　法
	File	可把文档、报表或其它类型的文件作为附件
	URL	任何有效的 URL 地址
	Snapshot	截图并保存为 JPG 格式的文件
	System Propertise	捕获和附上当前计算机的信息
	Clipboard Image	从剪贴板保存和附上图片

9.6　指定发布或周期的需求

在 Requirements 子模块定义需求后，在 Management 模块中创建周期，并向发布或周期分配需求。在 Requirement 模块，确定每个周期中需要覆盖哪些需求，并由此将这些需求分配到相关周期。使用需求决定程序中哪些内容需要测试。例如，在 Online Banking 程序中，需要测试用户是否能够登录他们的银行账户。针对这一功能，创建 Login 需求。Login 需求指定有效用户名和密码的相关需求。因此需要创建两个具体需求 Username 和 Password 作为 Login 需求的子需求。

在测试计划阶段，定义的需求是识别和创建测试的基础。例如，创建测试来测试 Username 和 Password 需求。并指定测试失败的条件。

当把需求分配给周期，需求自动地分配给发布。但是，如果把需求分配给周期，需求不会分配给发布中的其他周期。把需求分配给发布不会自动地将其分配给发布中的周期，分配需求到发布和周期，允许通过生成图表和报表来显示周期中覆盖发布和需求的质量和进程。

Test Lab 模块中的测试集文件夹包含需求覆盖的测试。在执行测试测试需求前，向周期分配测试集文件夹是用来：

1) 检查 Management 模块中的测试进度。

2) 确定发布或周期水平已解决的和突出的缺陷数量。

3) 强化测试集文件夹报告粒度。

如图 9-8 所示，当把测试集文件夹分配给周期，测试集文件夹中的所有测试自动地分配

给周期。测试集文件夹能够分配且只能分配给一个周期。

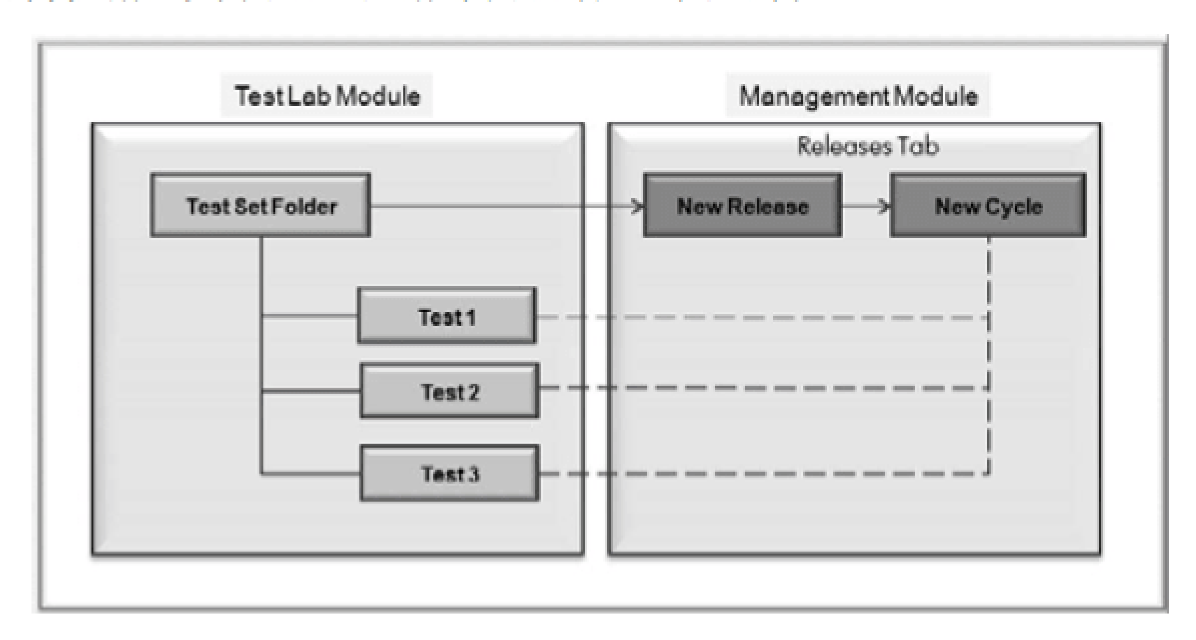


图 9-8　将测试分配给周期

9.7　进度标签

如图 9-9 所示，这个标签显示当前发布或周期进展状况的目测指标。可以看到已用时间和剩下时间，已经完成和剩下的测试实例，实际的和需要的执行比率等信息。

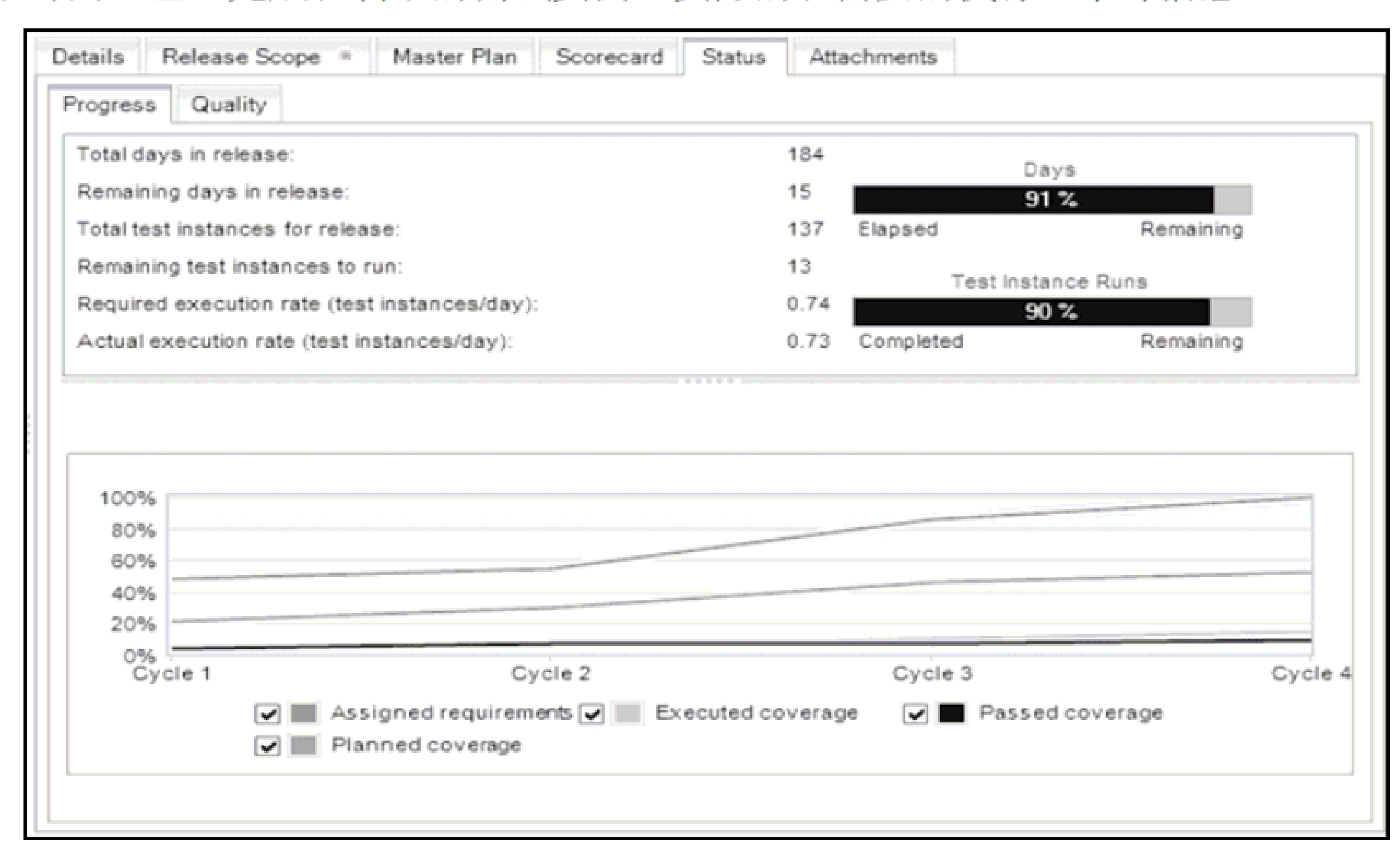


图 9-9　进度标签

9.8　质量标签

如图 9-10 所示，这个标签以图表形式显示了全过程中提交的发布或周期缺陷。你可以在应用程序管理流程的任何阶段检查发布的进展。

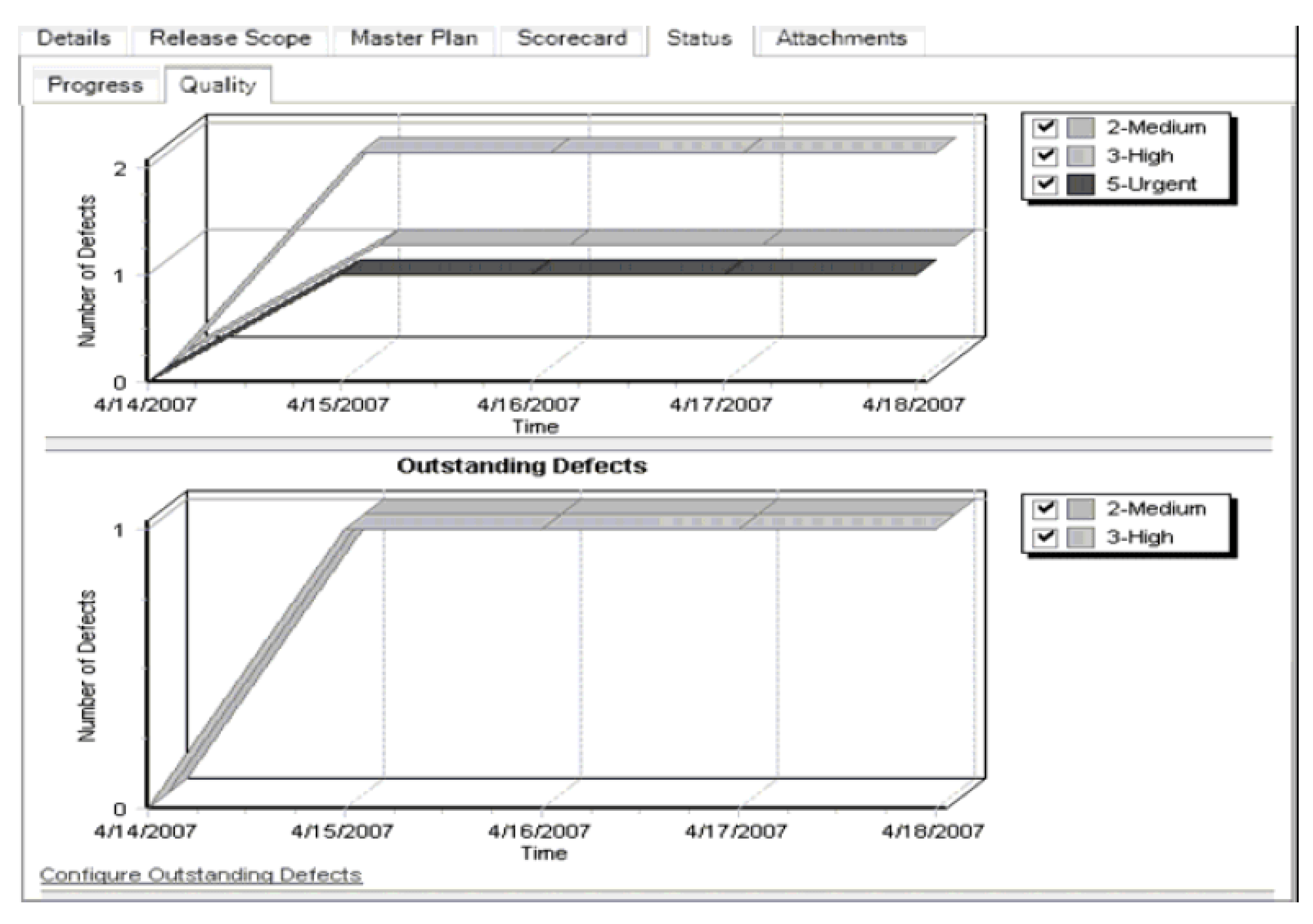


图 9-10　质量标签

习题与思考题

1. Cycle2 能在 Cycle1 结束之前开始吗?
2. 一个周期的结束日期能比发布的结束日期更晚吗?
3. 如何复制一个周期?

练习：创建发布树

如果你所在组织的软件开发团队已经完成了 Flight Reservation 程序的 release4.0 开发。现在需要测试改程序以验证产品新特点、现有职能和业务流程 Create Order 的实现。这些量度是以单独的测试周期环境完成的。总的可用测试时间是 90 天。

要测试应用程序，需要定义以下测试周期：

1. Cycle 1- New Features：这一周期测试 Flight Reservation 版本 4.0 的新特点。持续 30 天。

2. Cycle 2- Integrated System Test：本周期测试那些可能会影响现存功能的新特点变化。持续时间 10 天。

3. Cycle 3- Performance：一旦程序功能测试完成，周期要测量标准屏幕功能和业务流程，以确保新特点没有对反应时间造成消极影响。持续时间 10 天。

4. Cycle 4-User Acceptance Testing：本周期确保新特点满足业务期望。

在本次练习中，需完成以下任务：

第一部分：创建发布
第二部分：在发布中创建周期
第三部分：指定周期的详细内容
第四部分：重新安排日期
(该练习的指导步骤请参见本章正文)

第10章　需求管理

10.1　定义需求的优点

使用 ALM 的应用程序生命周期管理路线图包括以下阶段。

如图 10-1 所示，在整个应用程序生命周期中，可以通过生成报告和图来监视和控制策略点。

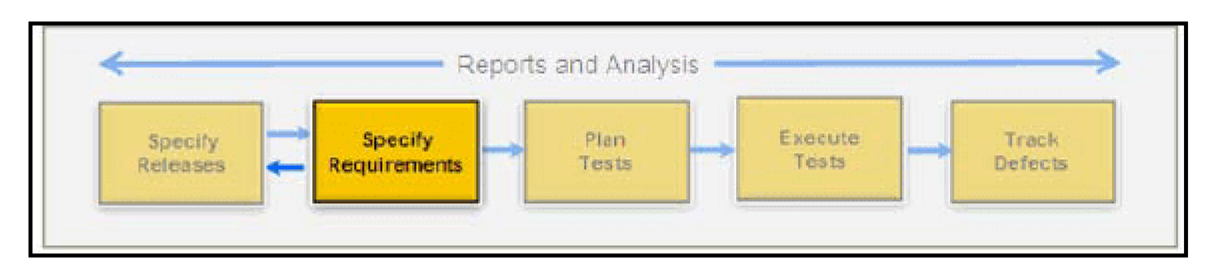


图 10-1　ALM 路线图

“需求”详细描述需要解决或实现的内容，以达成正在开发的应用程序的目标。在项目前端清晰正确地定义需求提供以下优点：

1) 向利益相关者提供定义优先级的指导方针。
2) 在利益相关者之间设定清晰的预期。
3) 减少浪费并消除不必要的支出。

需求模块允许你在应用程序生命周期管理的各个阶段定义、管理和跟踪需求。

10.2　介绍需求

此任务描述如何在 ALM 中创建和管理需求。

此任务包括以下步骤：

1) 先决条件
2) 创建需求
3) 导入业务流程模型
4) 跟踪需求
5) 计算风险
6) 创建覆盖率
7) 链接到缺陷
8) 分配至版本
9) 分析需求
10) 建立基线

对于每个需求主题，QA 测试员均应该创建相应的详细测试需求列表。例如，Application Security 需求主题可能会被分解为如图 10-2 所示的需求。



图 10-2　需求主题

在需求树中的每一个需求均要求被详细描述，并且应该包括所有与需求相关的附件。QA 测试人员分配每个需求一个优先级，此优先级会作为测试组创建测试计划的一个考虑因素。

1. 先决条件

通过收集功能和技术规范、市场和业务需求文档以及利益相关者目标等信息，确定需求的范围。可能会提出一些问题：

1) 应用程序的主要目的和方向是什么？
2) 应用程序的临界约束是什么？
3) 应用程序的主要功能是什么？
4) 应用程序功能中每个元素的相关重要性是什么？
5) 应用程序的严重或高风险功能是什么？
6) 业务或测试优先级是什么？
7) 客户/最终用户是否同意你设定的优先级？
8) 总体质量目标是什么？

2. 创建需求

通过创建需求树，定义需求范围的层次结构框架。在需求树中定义不同需求组。对每个需求组，在需求树中创建详细需求的列表。树中的每个需求都可以包括任何相关附件和多信息文本文档。然后为需求分配优先级，在创建测试计划时，可能会考虑此优先级。

3. 导入业务流程模型

如果使用业务流程模型，可以通过导入使用标准建模工具创建的模型，创建需求的框架。“业务模型”模块允许分析业务流程模型和业务流的质量。

4. 跟踪需求

可在需求之间添加可跟踪性。分析在特定需求中建议更改的影响时，可跟踪性会显示可能受此更改影响的其他需求。要确定需求之间关系的完整性，可生成可跟踪性矩阵。

5. 计算风险

可根据需求的性质和你掌握的资源，使用基于风险的质量管理计算在哪个级别，测试每项需求。

6. 创建覆盖率

在需求和测试之间创建覆盖率，以确保在项目中实现所有需求。 也可以通过在测试计划树中将需求转换到测试，来创建覆盖率。覆盖率在需求及其对应测试之间自动创建。

7. 链接到缺陷

可将需求链接到特定缺陷。在需求和测试之间创建覆盖率时，这很有用。链接缺陷有助于确保符合需求和测试需要。如果需求更改，可以立即识别出哪些测试和缺陷受到影响，谁应对此负责。

8. 分配至版本

将需求分配到在“版本”模块的版本树中定义的版本或周期。

(1) 分配至版本。右键单击需求，选择分配至版本。

(2) 分配至周期。右键单击需求，选择分配至周期。

9. 分析需求

复审需求以确保它们满足定义的需求范围。在批准需求之后，将需求的状态从未审阅更改为已审阅。为帮助复审需求，可以生成报告和图。

10. 建立基线

创建基线以批准或比较应用程序生命周期中的重要里程碑。

10.2.1　需求树

定义测试需求的过程是单调乏味并耗时的。大多数情况下，测试组是用字处理软件或电子表格文档来记录需求。这样的需求文档维护复杂，难以组织和分类，而且不能连接到基于需求创建的测试和相应的缺陷。

作为整个测试过程的第一步，ALM 需求模块能够让你简单地定义和管理你的测试需求。测试小组通过在 ALM 中创建需求树来记录需求。它以图表方式展现整个需求说明和需求关系。一旦创建了测试，就可以在需求和测试间建立连接，而测试与缺陷又可以建立关联。通过这种方式可以跟踪测试需求到整个测试过程的每一步骤。假如测试需求被改变，你可以立即确定哪些测试和缺陷受到影响，并且是谁负责的受到了影响。你能在需求树中分组和分类需求，监控需求过程和任务分配，并产生详细的报告和图表。如图 10-3 所示。

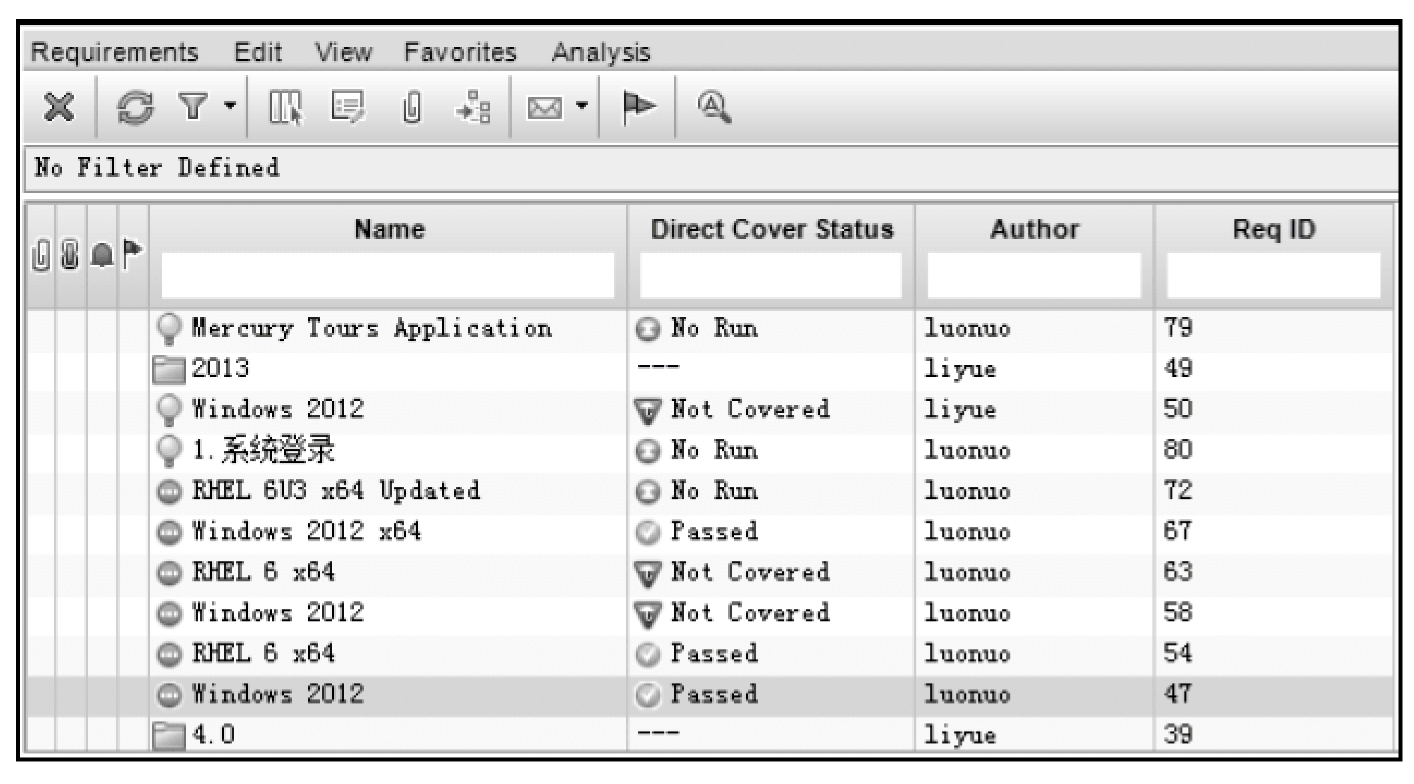


图 10-3　需求树

注意：本章假定你使用需求模块的 Document View 去创建需求树。

ALM 在需求树中有机地组织并显示数据。需求树中每一行都显示了一条独立的需求。需求树中可以显示如表 10-1 所示细节信息。

表 10-1　需求树细节信息

选　项	描　述
作者	创建此需求的用户名。默认情况，ALM 将登录用户名插入到此字段
创建日期	需求被创建的日期。默认情况下，创建日期被设置为当前服务器日期。你也可以单击下拉箭头去显示一个日历，并选择一个不同的创建日期
创建时间	需求被创建的时间。默认情况下，创建时间被设置为当前服务器的时间
描述	需求描述
覆盖状态	需求当前的状态。默认情况下，状态为 Not Covered。 一个需求的状态能够是如下几种： Not Covered：这个需求没有被链接到测试。 Failed：覆盖此需求的一个或多个测试被执行，且状态为：Failed。 Not Completed：覆盖此需求的一个或多个测试被执行，且状态为：Not Completed。 Passed：覆盖此需求的所有测试均有同样状态：Passed。 No Run：覆盖此需求的所有测试均有同样状态：No Run。 可单击一下 State，去打开你所选择需求的测试覆盖对话框。 N/A：不适用
修改	标识此需求被最后修改的时间
名称	需求名
优先级	需求的优先级。范围从最低级别(Level 1)到最紧急级别(Level 5)
产品	需求所基于的应用程序组件
需求 ID	需求的唯一数字 ID，由 ALM 自动分配。注意，需求 ID 是只读的
复查	标识此需求是否被复查，并且被责任人批准通过
类型	需求的类型，可以是 Hardware 或 Software
附件	指示本需求是否包含附件

10.2.2　需求规范及类型

通过创建需求树，在需求模块中记录需求。需求树是需求规范的图形表示，显示不同需求之间的层次结构关系。树包括基于需求类型或功能区域的不同需求组。

对每个需求组，在需求树中创建详细需求的列表。详细描述树中的每个需求，并可以包括任何相关链接和附件。创建需求树后，需求就可用作在测试计划树中定义测试的基础。

描述需求目的(比如功能需求或测试需求)的需求类型(如表 10-2 所示)。将每个需求分配到 ALM 默认需求类型之一，或将其分配到项目管理员可创建的自定义需求类型。

表 10-2　需求类型

需求类型	描　　述
	业务：业务流程需求。默认情况下，不能将覆盖率添加到此需求
	文件夹：用于组织需求的文件夹。默认情况下，不能将覆盖率添加到此需求
	功能：系统行为需求
	组：相关需求的集合
	测试：系统性能需求
	业务模型：表示业务流程模型实体的需求
	未定义：未定义的需求

10.2.3　使用需求树视图

图 10-4 允许在树中按层次结构查看需求。

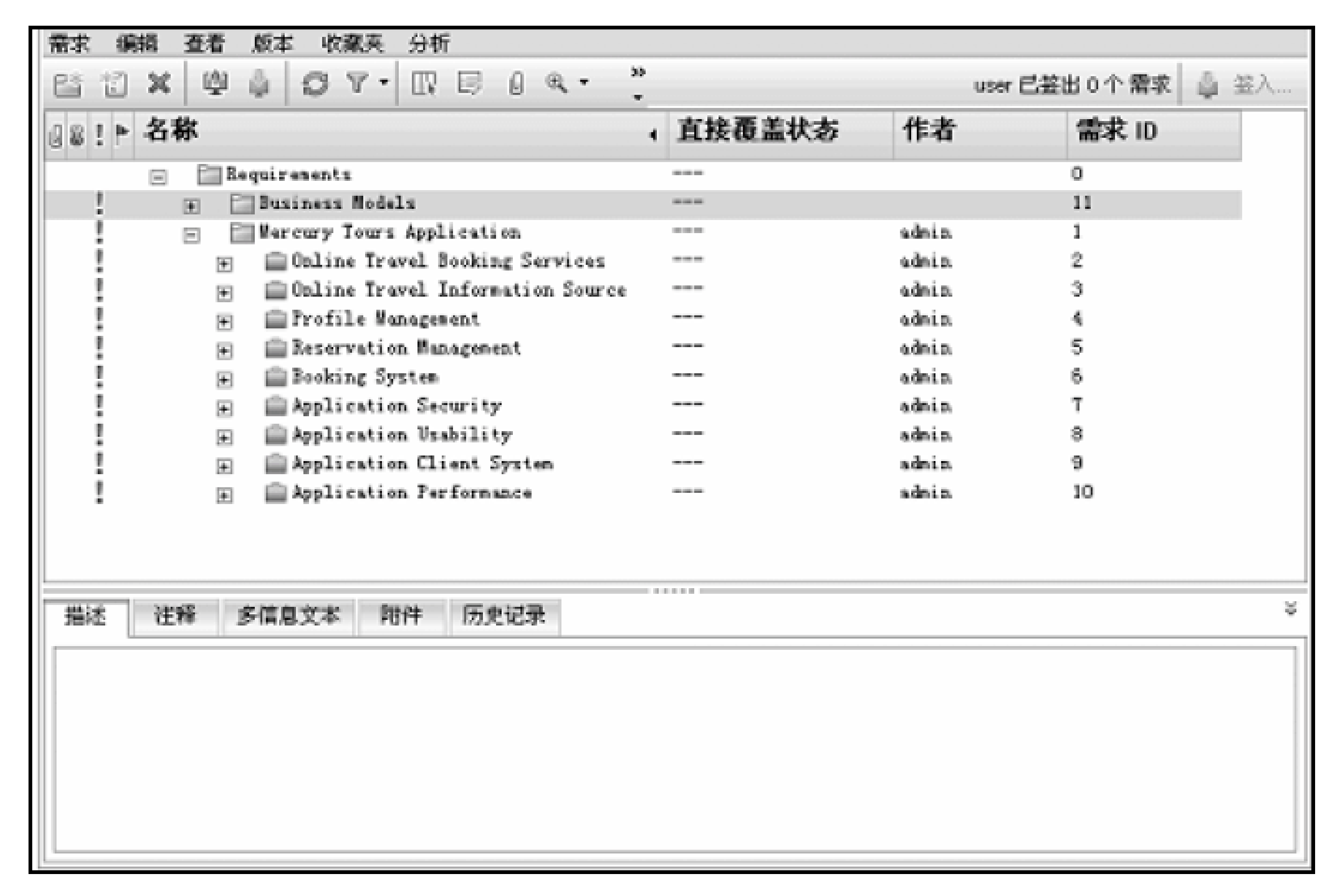


图 10-4　需求树视图

10.3　创建需求树

需求工具栏包括如下的按钮：

注：以下内容为按图标的顺序依次介绍创建需求树的工具按钮。如图 10-5 所示。



图 10-5　创建需求树的工具

1) New Requirements：新建需求，增加一个新的需求到需求树。ALM 将增加此需求到当前所选择的需求下面，并处于相同等级。

2) New Child Requirements：新建子需求，增加一个新的需求到需求树。ALM 将增加此子需求到当前所选择的需求下面，并处于低一级的级别。

3) Delete：删除，从需求树中删除所选择的需求。

4) Refresh Selected：刷新，刷新在需求模块中的数据。

(1) 单击 Refresh Selected 按钮，刷新当前所选择的需求。所有子需求也会被同时刷新。

(2) 单击箭头并选择 Refresh All 去刷新所有的需求。

5) Select Columns：选择列，打开选择列对话框，你可以决定哪些字段显示在需求树中，并决定它们的显示顺序。

6) Zoom in：展开，改变需求树的细节等级。

(1) 单击 Zoom In 按钮展开需求树的指定分支。

(2) 单击 Zoom In 箭头并选择 Zoom Out One Level 去取消预先展开的命令。

(3) 单击 Zoom In 箭头并选择 Zoom Out To Root 去收缩，并显示整个需求树的根结点。

7) Find：查找，打开查找需求对话框，能够让你在需求树中查找你想要的需求。

8) Mail Requirement：发送需求，打开发送邮件对话框，可从邮件列表中选择收件人，或输入其他邮件地址，发送需求邮件。

9) Attachments：附件，打开附件对话框，为所选的需求添加附件。

10.4　创建需求

此任务描述如何定义和更新“需求”模块中的需求。在创建需求树后，可将需求用作在测试计划树中定义测试的基础。

注：此任务是较高级别任务的一部分。

此任务包括以下步骤：①“创建需求”；②“导入需求”；③“更新需求”；④“将需求转换到测试”。

1. 创建需求

1) 打开“需求”模块。在 ALM 侧栏的需求下，选择需求。在查看菜单中，选择需求树。

2) 创建文件夹。右击需求根文件夹，然后选择新建文件夹。要创建子文件夹，请右击文件夹并选择新建文件夹。

3) 添加需求。右击需求文件夹，选择新建需求。要创建子需求，请右击需求并选择新建需求。

2. 导入需求(可选)

除了直接在 ALM 中创建需求以外，还可以从 Microsoft Word、Microsoft Excel 或其他第三方需求管理工具将需求导入 ALM 项目。要导入需求，必须先安装相应的插件。

3．更新需求

对于每个需求，可以更新其详细信息、附件和多信息文本文档。右击需求，选择需求详细信息。将打开“需求详细信息”对话框。

4．将需求转换到测试

为帮助在“测试计划”模块中建立测试计划树，可将需求用作定义测试的基础。可以重用需求，并在测试计划树中将它们转换为以下实体：测试主题、测试、测试步骤或步骤描述。

根据需求创建测试的方法有以下两种：

1) 将需求转换到测试。使用此选项可将需求转换到测试主题、测试、测试步骤或步骤描述。右键单击需求或文件夹，选择转换到测试。将打开“转换到测试”向导。

2) 将需求转换到测试，并将测试添加到测试集。使用此选项可将需求转换到测试计划树的指定主题中和“测试实验室”模块的指定测试集中的测试。右键单击需求，选择生成测试。将打开“生成测试”对话框。

10.4.1　如何创建需求——用例场景

图 10-6 用例场景提供在“需求”模块中指定需求的示例。以下场景使用基于应用程序的项目预订航班和度假。可将主要需求定义为：Online Travel Booking Services、Online Travel Information Source、Profile Management、Reservation Management、Booking System、Application Security、Application Usability、Application Client System 和 Application Performance。

对每个需求组，在需求树中创建详细需求的列表。Profile Management 需求可细分为图 10-7 所示需求。

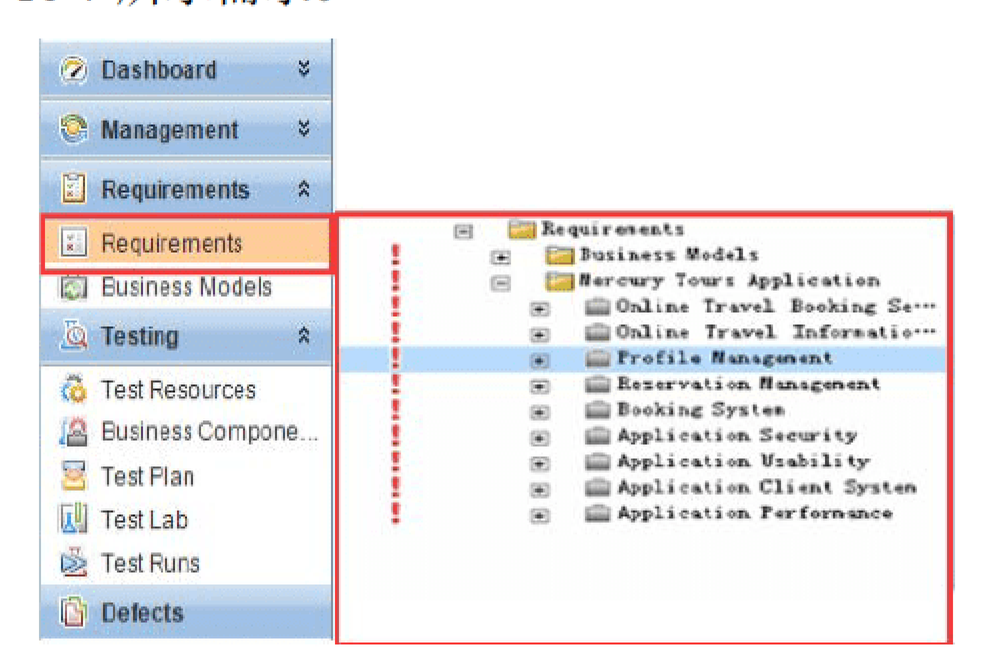


图 10-6　场景示例一



图 10-7　场景示例二

树中的每个需求都可以包括任何相关附件和多信息文本文档。

10.4.2　需求详细信息

图 10-8 显示了 Requirements 视图。

此页面允许你更新任何需求的详细信息、附件、测试覆盖率、需求可跟踪性链接、基于风险的质量管理设置和缺陷链接。还可以查看对需求所做更改的列表。见图 10-9 需求详细信息。

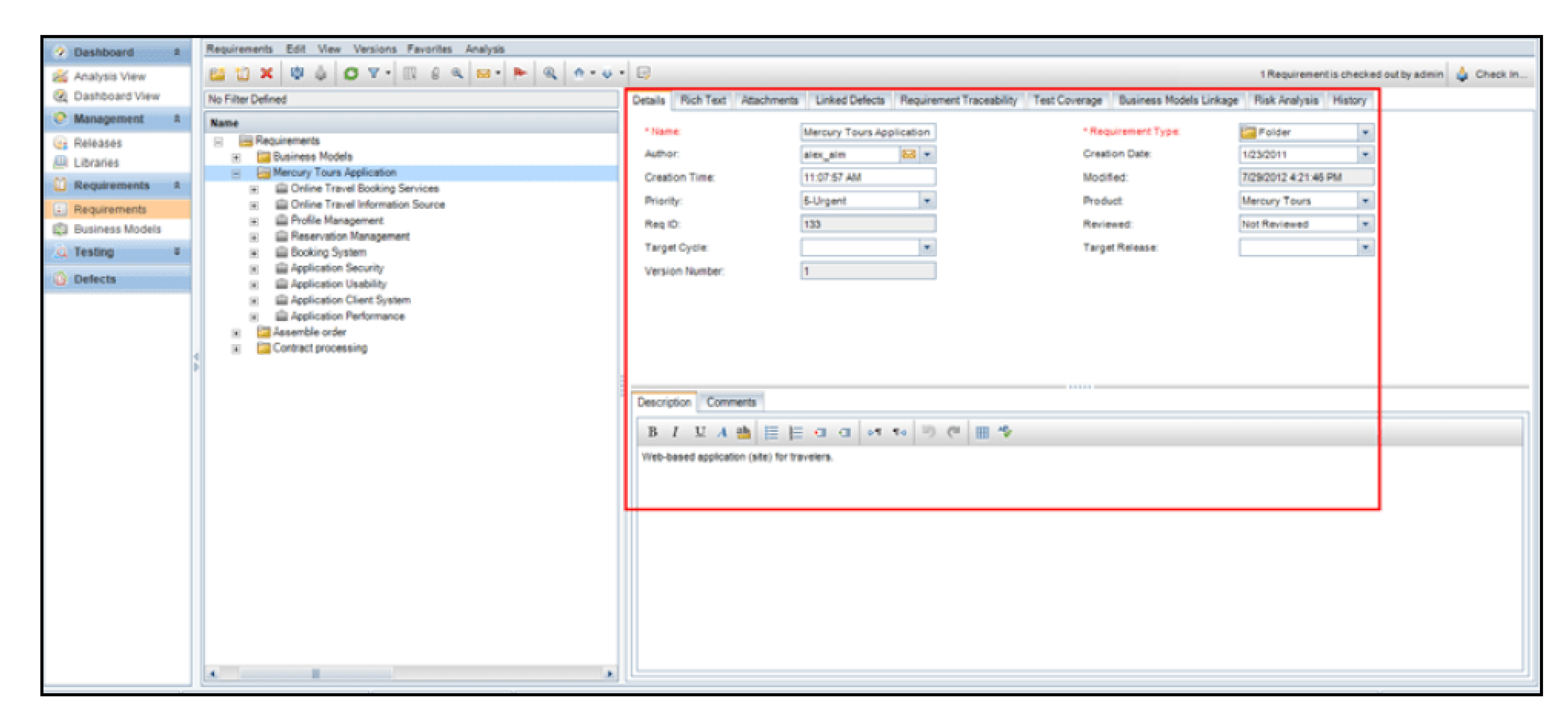


图 10-8　需求视图

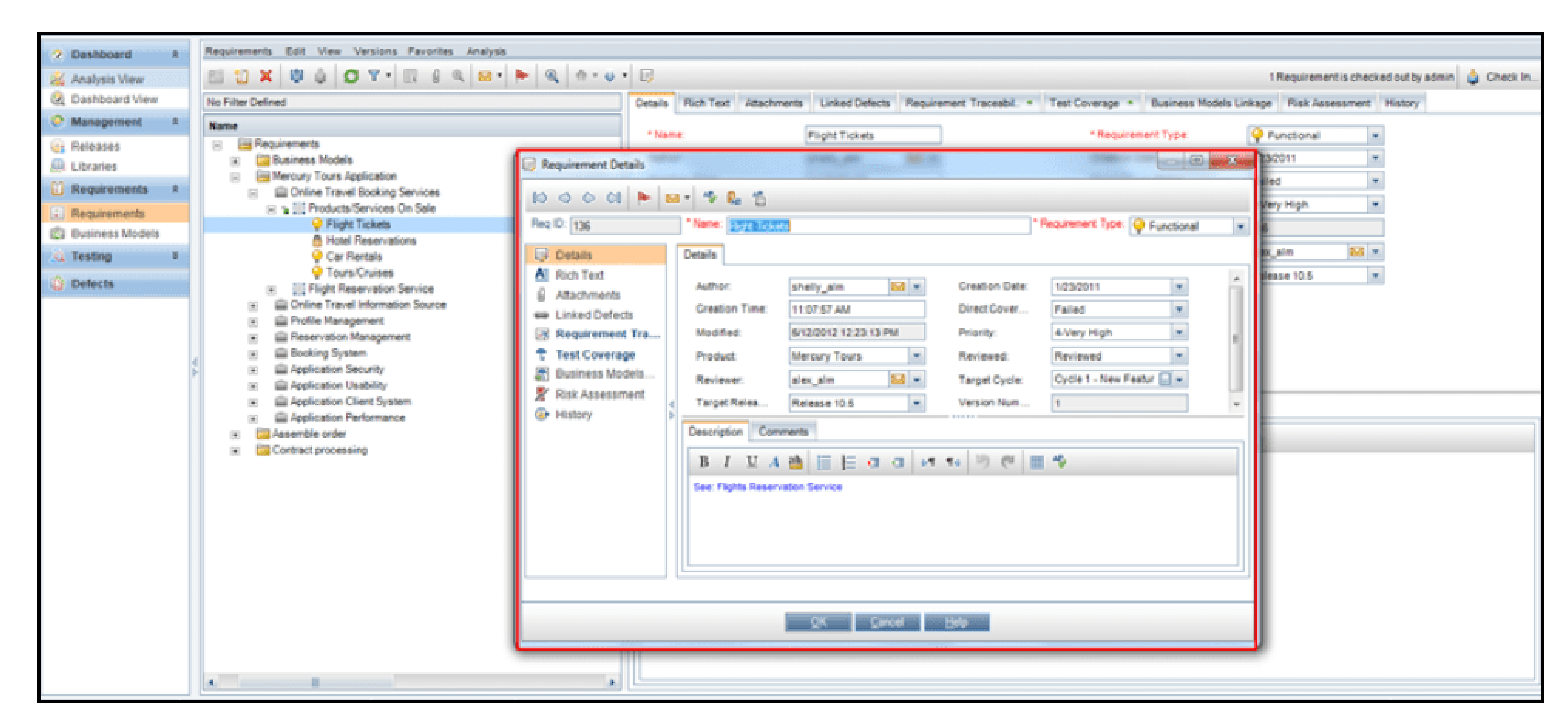


图 10-9　需求详细信息

10.4.3　需求模块菜单和按钮

此部分描述在“需求”模块中可用的菜单和按钮(参见表 10-3 和表 10-4)。

表 10-3　描述需求模块

访问	在 ALM 侧栏上的需求下面，选择需求
重要信息	某些菜单命令和按钮并非在所有需求视图中都可用。 版本控制：在启用了版本控制的项目中，有额外的菜单命令和按钮可用。 Quality Center Starter Edition ：并非所有命令和按钮都可用

用户界面元素如下所述(无标签元素显示在尖括号中)：

表 10-4　需求模块菜单和按钮

UI 元素(A～Z)	菜　单	描　述
添加到收藏夹	收藏夹	打开“添加收藏夹”对话框，允许你将收藏夹视图添加到专用或公用文件夹中
警报	编辑	打开“警报”对话框，允许显示该需求的警报
分配至周期	需求	打开“选择周期”对话框，允许将选择的需求分配至周期
分配至版本	需求	打开“选择版本”对话框，允许将选择的需求分配给版本
附件	<右键单击菜单>	打开“附件”页面，允许你将附件添加到选定需求
清除警报	编辑	清除该模块已显示的警报
清除后续标志	编辑	清除选定需求的后续标志
折叠	查看	在需求树中折叠所有分支
配置可跟踪性矩阵	编辑	打开“配置可跟踪性矩阵”向导，允许你设置矩阵中显示的实体。对以下项可用：可跟踪性矩阵
需求转换到测试	需求	打开“转换到测试”向导，允许你将选定需求转换到测试计划树的指定主题中的测试
复制/粘贴	编辑	在相同项目中或跨项目复制选定需求。复制需求时，还将复制该需求的任何子项。 注： 不会复制需求的测试覆盖率、缺陷链接和基于风险的质量管理数据。 要复制具有可跟踪性的需求，还必须复制其关联的跟踪需求。 不能在同一项目内复制根文件夹。 如果粘贴与现有需求具有相同名称的需求，则后缀_Copy 将自动添加到同名需求的名称的末尾
复制 URL/ 粘贴	编辑	复制选定需求，并粘贴其 URL 作为链接。不复制需求自身。但可以将地址粘贴到另一个位置中，比如电子邮件或文档。单击链接将打开 ALM，并将你带到该需求。如果尚未登录，则 ALM 将提示你提供登录详细信息
覆盖率分析	查看	显示“覆盖率分析”视图，允许你根据测试覆盖率状态分析子需求的细分
剪切/粘贴	编辑	在需求树中将选定需求移动到不同位置。 注：在需求树中将需求移动到不同位置时，还会移动其子需求、测试覆盖率、需求可跟踪性链接和缺陷链接。无法移动根文件夹。 提示：通过拖动需求，也可以将需求移动到需求树中的新位置

(续表)

UI 元素(A～Z)	菜　单	描　述
删除	编辑	删除选定需求。删除需求还会删除其子需求、测试覆盖率、需求可跟踪性链接和缺陷链接。无法删除根文件夹。 版本控制：删除需求时会删除需求的所有以前版本
全部展开	查看	在需求树中展开所有分支
导出	需求	打开“导出覆盖率分析”对话框，允许你将“覆盖率分析”视图保存到 Microsoft Word 文档。文档会针对需求树中的每个父需求显示一张条形图。 对以下项可用：覆盖率分析
导出	<右键单击菜单>	打开“导出所有网格数据”对话框，允许你将网格中的需求导出为文本文件、Microsoft Excel 工作表、Microsoft Word 文档或 HTML 文档。 选择以下某个选项： 全部。导出网格中的所有需求。 已选择。导出网格中的选定需求。 对以下项可用：需求网格
生成可跟踪性矩阵	分析	打开“生成可跟踪性矩阵”对话框，允许将“可跟踪性矩阵”视图保到 Microsoft Excel 工作表。 对以下项可用：可跟踪性矩阵
筛选器/排序	查看	允许你筛选和排序需求树或网格中的需求
查找	编辑	在“需求”模块中搜索需求
查找下一项	编辑	搜索需求网格中符合先前定义的搜索条件的下一项。 对以下项可用：需求网格
后续标志	编辑	打开“后续标志”对话框，允许你定义选定需求的后续标志
生成测试	需求	打开“生成测试”对话框，允许将选定需求转换到测试计划树的指定主题中和“测试实验室”模块的指定测试集中的测试
转至需求	需求	打开“转至需求”对话框，允许你按需求 ID 查找特定需求。要在需求树中显示需求，请单击以树视图显示。要在“需求详细信息”对话框中显示需求，请单击显示详细信息。只能转到当前筛选器中的需求
转到需求树中的需求	<右键单击菜单>	转到“需求树”视图，突出显示选定需求。 对以下项可用：需求网格和可跟踪性矩阵
图	分析	列出可以针对需求数据生成的图。选择预定义图，或启动图向导
网格筛选器	查看	在每个列名下方显示网格筛选器框，允许你定义列的筛选器条件。直接输入框中，或单击该框以显示“浏览”按钮，这将打开“选择筛选器条件”对话框。 对以下项可用：需求网格

(续表)

UI 元素(A～Z)	菜　单	描　述
指示器列	查看	有关更多详细信息，请参见“需求模块图标”
信息面板	查看	在需求视图的底部显示信息面板选项卡
反选	编辑	取消选择网格中所有先前选择的需求，并选择所有先前取消选择的需求。 对以下项可用：需求网格
下移 上移		允许你在需求树中上下移动选定需求以设置其顺序。 注：如果已使用“筛选器”对话框中的“查看顺序”选项卡对需求树中的需求排序，则上移和下移按钮不可用。 对以下项可用：需求树、需求详细信息和覆盖率分析
新建文件夹	需求	打开“创建新需求文件夹”对话框，可用于在选定文件夹下面添加文件夹。 对以下项可用：需求树、需求详细信息和覆盖率分析
新建需求	需求	打开“新建需求”对话框，允许你在选定需求下面添加需求。 对以下项可用：需求树、需求详细信息和覆盖率分析
组织收藏夹	收藏夹	打开“组织收藏夹”对话框，使你能够通过更改属性或删除视图来组织收藏夹视图的列表
专用	收藏夹	列出仅创建收藏夹视图的用户才可访问的收藏夹视图
公用	收藏夹	列出所有用户都可访问的收藏夹视图
最近使用	分析	在“需求”模块中显示最近查看的报告和图
全部刷新	查看	刷新需求树或网格使之显示最新需求
重命名	编辑	重命名选定需求。无法重命名根文件夹。语法异常：需求名称不能包括以下字符：\ ^ *
替换	编辑	替换树或网格中的需求字段值
报告	分析	列出可以生成需求数据的预定义报告
需求详细信息	需求	打开“需求详细信息”对话框，可用于显示所选需求的详细信息
需求详细信息	查看	显示“需求详细信息”视图，允许你在需求和其他实体之间创建链接。它还可用于计算和分析需求风险
需求网络	查看	显示“需求网格”视图，允许你在平面非层次结构视图中查看需求。网格中的每行都显示一条单独的需求
需求树	查看	显示“需求树”视图，允许你在树中按层次结构查看需求
全选	编辑	选择网格中的所有需求 对以下项可用：需求网格

（续表）

UI 元素(A～Z)	菜　单	描　述
选择列	查看	打开“选择列”对话框，允许你确定需求树或网格中显示哪些字段及其显示顺序
通过电子邮件发送	需求	打开“发送电子邮件”对话框，允许你将需求电子邮件发送给从列表选择的接收方或需求的作者
显示完整路径	查看	显示需求树中需求的路径。 对以下项可用：可跟踪性矩阵
测试覆盖率 > 显示向覆盖率添加条件	查看	启用“添加条件覆盖率”对话框。 对以下项可用：需求树、需求详细信息和覆盖率分析
文本搜索	编辑	打开“需求”模块窗口下半部分中的文本搜索窗格，允许你在预定义的字段中搜索记录
更新选定项	编辑	打开“更新选定项”对话框，允许你更新树或网格中多个选定需求的字段值
缩放	查看	在需求树中更改详细信息的级别。包括以下选项：放大。显示选定需求，并包括带有需求的层次结构路径的标题。 缩小一级。撤消以前的放大命令。缩小至根目录。放大到最高层次结构级别，并显示整个需求树
<最近使用的收藏夹视图列表>	收藏夹	显示“需求”模块中四个最近使用的收藏夹视图。可通过在“站点管理”中的站点配置选项卡中设置 FAVORITES_DEPTH 参数，定义在菜单上显示的视图数

10.4.4　描述和注释选项卡

图 10-10 所示选项卡允许你定义需求之间的关系。它以网格结构显示，有助于你理解需求之间存在的关联和依赖关系。

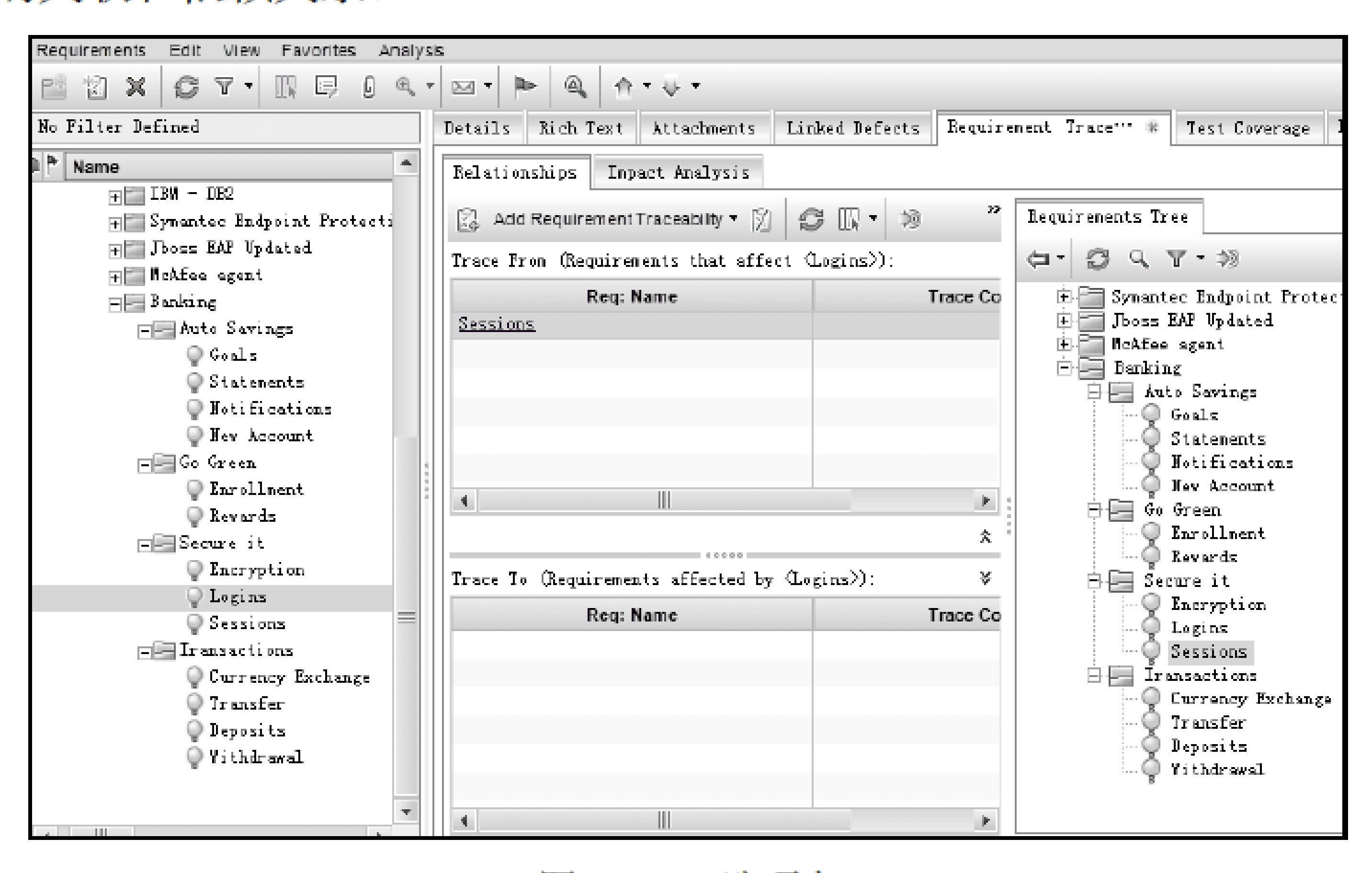


图 10-10　选项卡

10.4.5　查看需求历史

查看需求的修改历史：

1) 在需求树上选择一个需求。

2) 选择 View | Description & History。

3) 单击 History 标签页，所有字段的修改历史都显示在网格中。

对于需求的每一个改变，网格上都会显示相应的修改日期、修改人名称和修改后的值，如图 10-11 所示。你能够定义哪些字段可以显示在网格中。在 Field 列表中，选择一个字段名，仅查看该字段的修改记录。

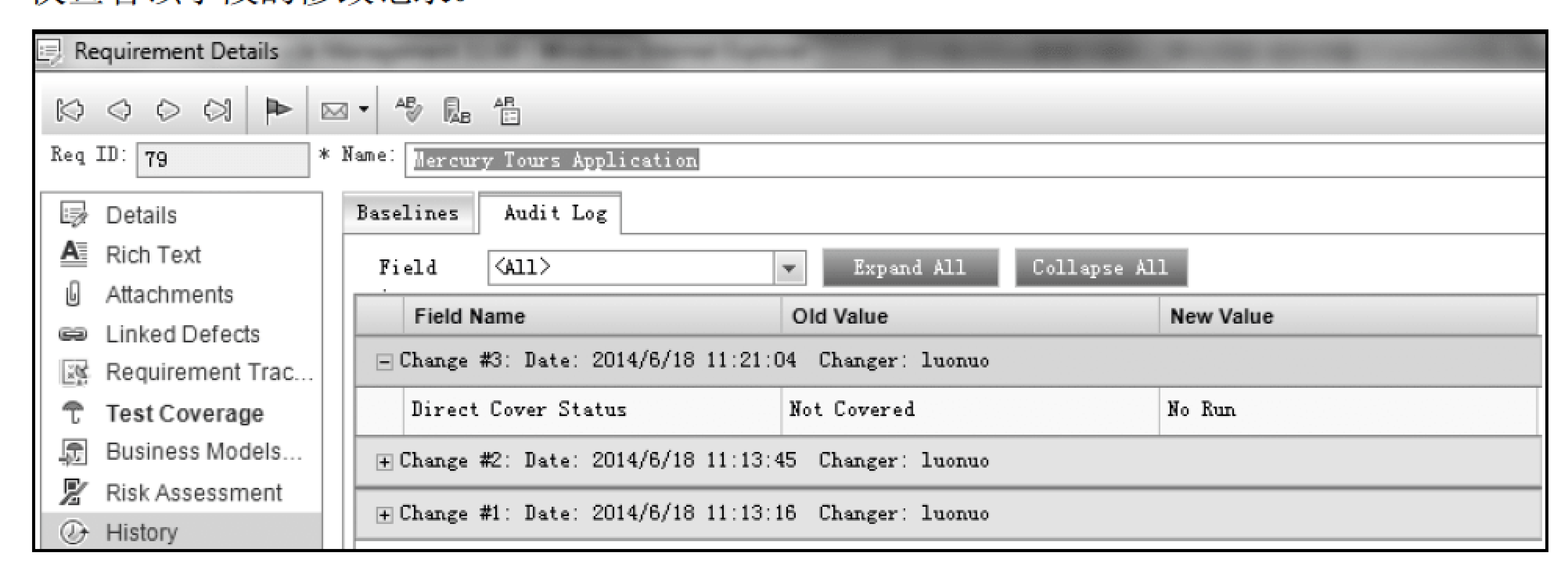


图 10-11　查看需求历史

10.4.6　需求网格

图 10-12 视图允许你在平面非层次结构视图中查看需求。网格中的每行都显示一条单独的需求。

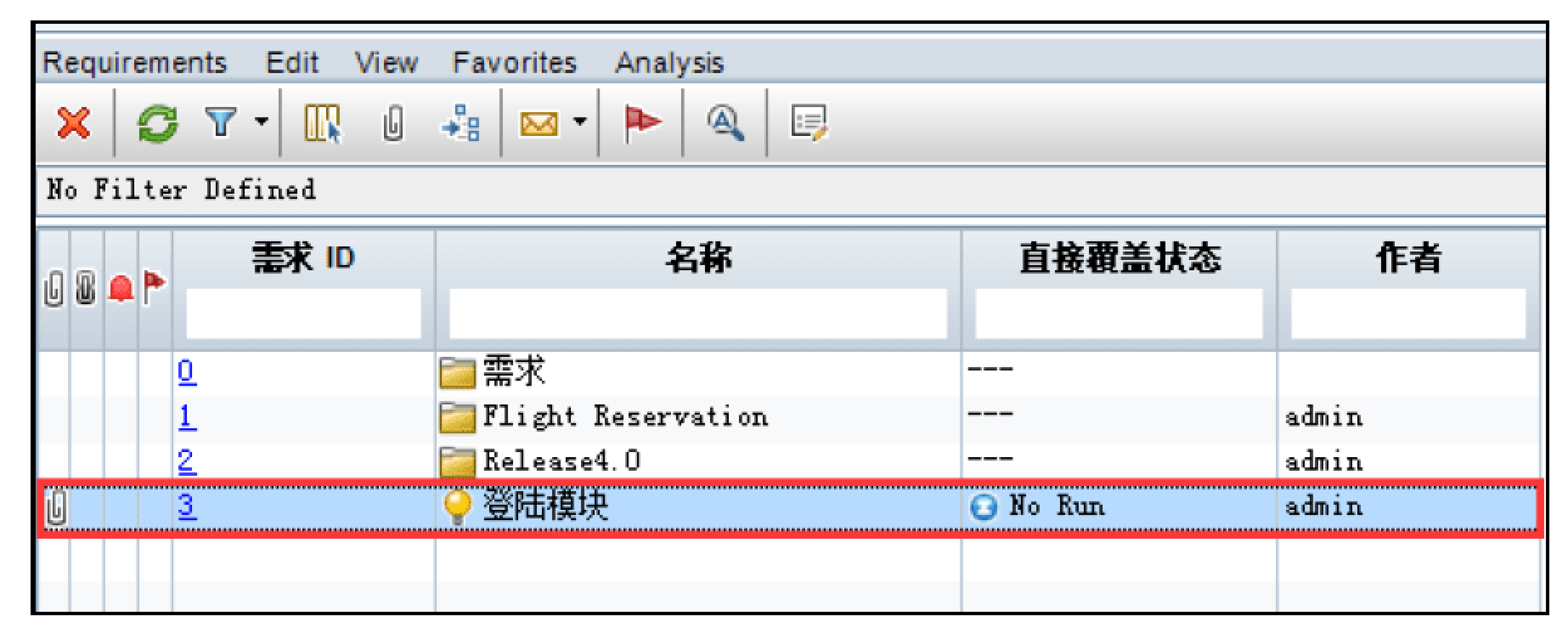


图 10-12　网格视图

10.5　需求模块

10.5.1　需求模块

在 ALM 中单击左侧工具栏中的 Requirements 定义需求。你可以使用下面 4 种方式显示需求，如图 10-13 所示。

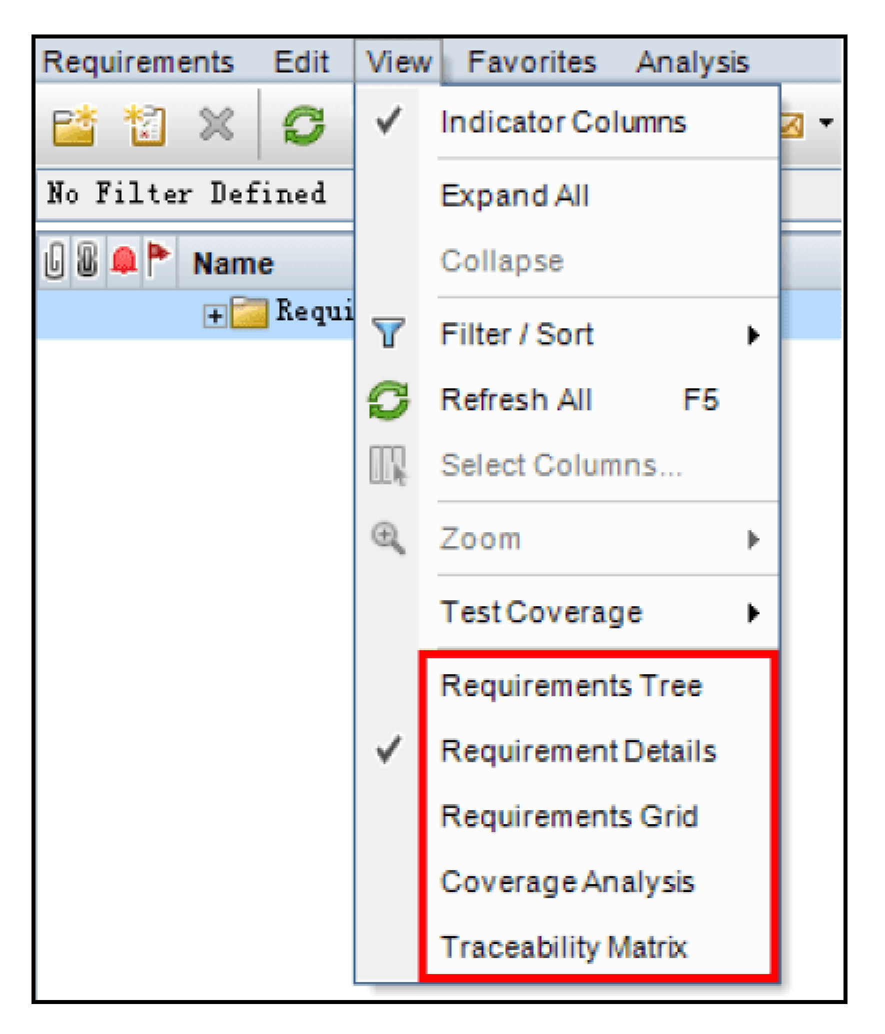
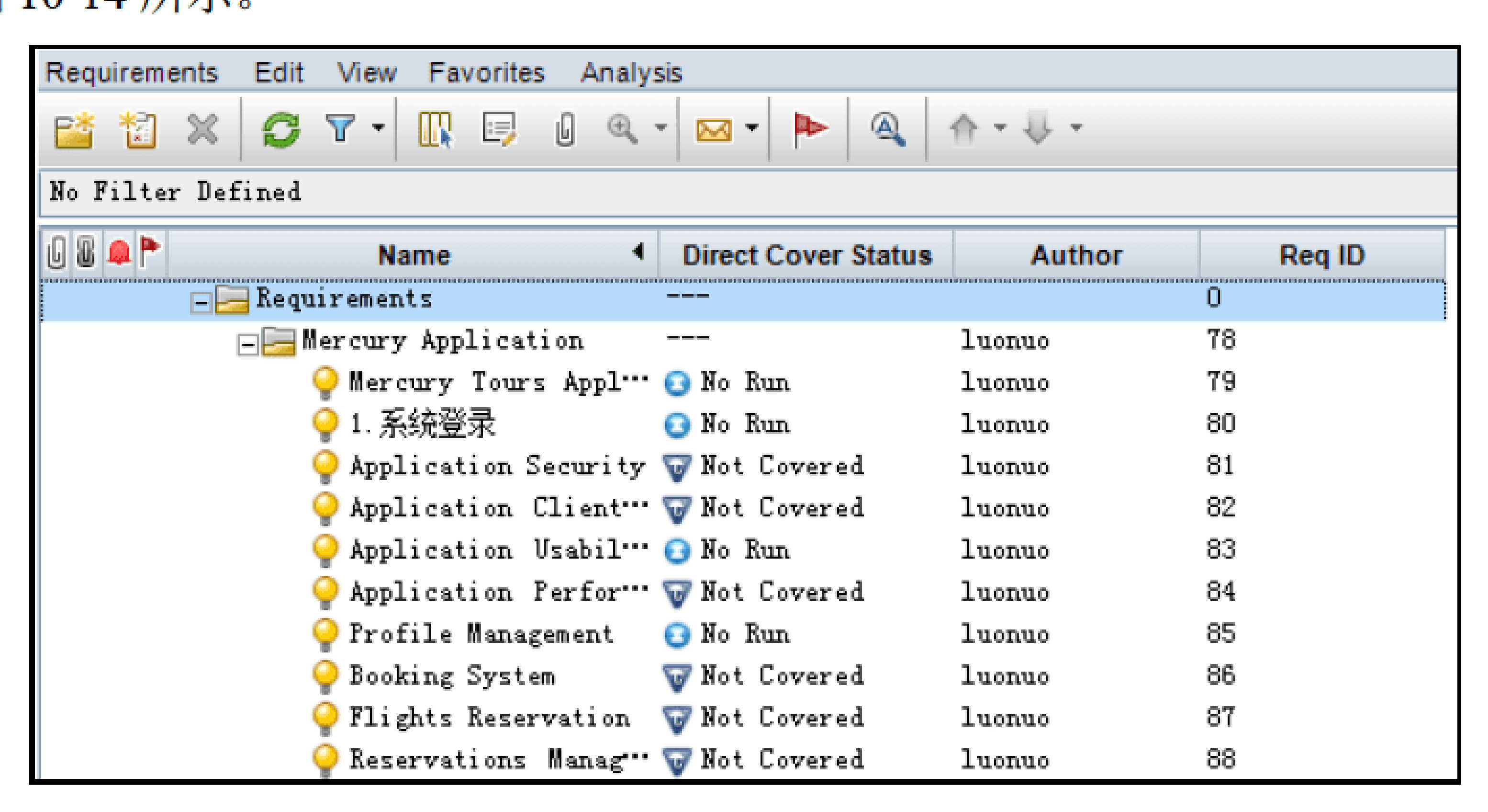


图 10-13　需求的 4 种显示方式

注意：可从 Microsoft Word、Excel 或第三方的需求管理工具中将需求导入到你的 ALM 工程。对于导入需求，你必须首先安装相应的 ALM 插件。对于更详细信息请查看《ALM 安装指南》。

1) Requirement Tree

如图 10-14 所示。

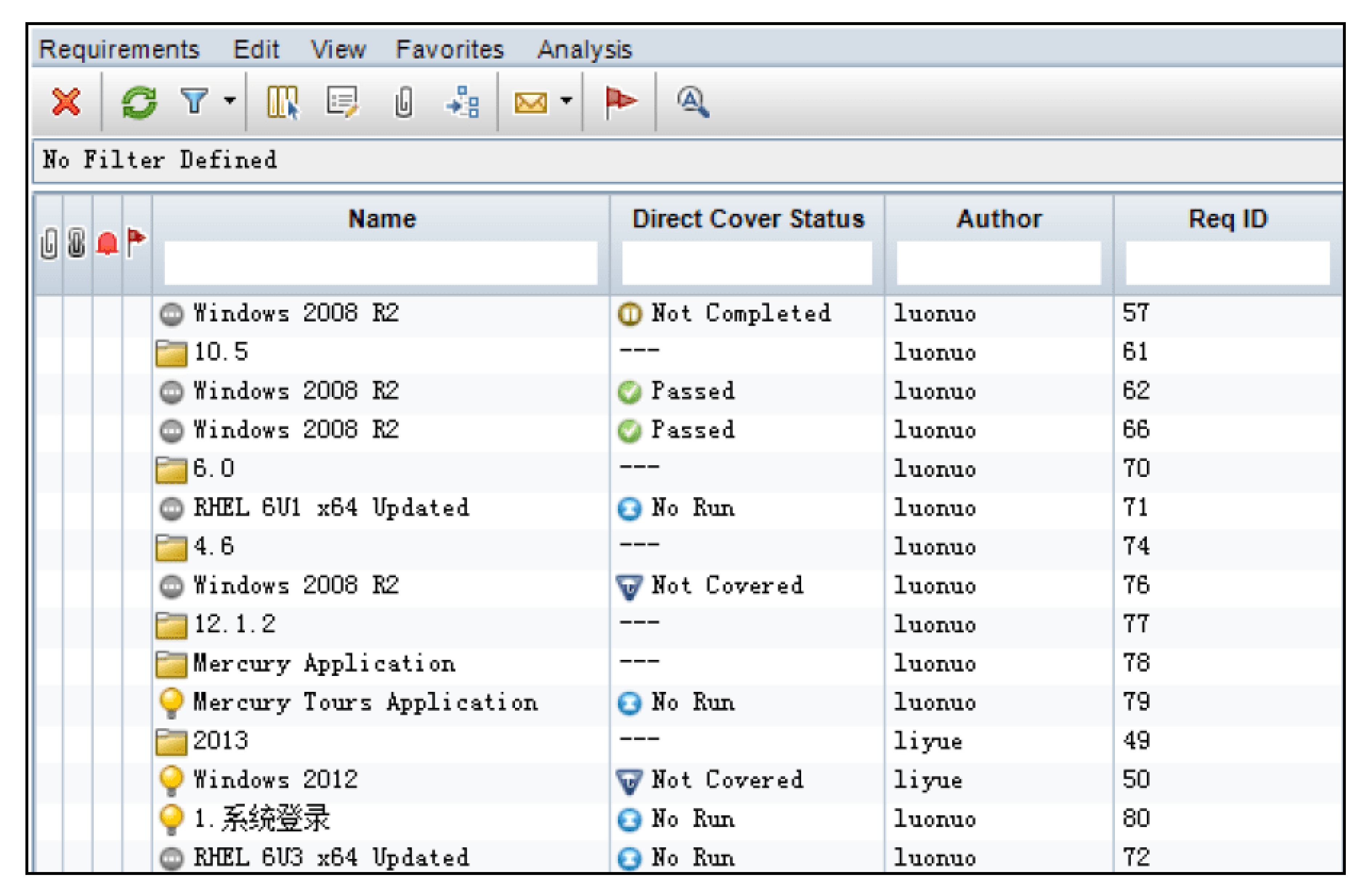


Name	Direct Cover Status	Author	Req ID
Requirements	---		0
Mercury Application	---	luonuo	78
Mercury Tours Appl…	No Run	luonuo	79
1. 系统登录	No Run	luonuo	80
Application Security	Not Covered	luonuo	81
Application Client…	Not Covered	luonuo	82
Application Usabil…	No Run	luonuo	83
Application Perfor…	Not Covered	luonuo	84
Profile Management	No Run	luonuo	85
Booking System	Not Covered	luonuo	86
Flights Reservation	Not Covered	luonuo	87
Reservations Manag…	Not Covered	luonuo	88

图 10-14　需求树显示方式

2) Requirements Grid

如图 10-15 所示。



Name	Direct Cover Status	Author	Req ID
Windows 2008 R2	Not Completed	luonuo	57
10.5	---	luonuo	61
Windows 2008 R2	Passed	luonuo	62
Windows 2008 R2	Passed	luonuo	66
6.0	---	luonuo	70
RHEL 6U1 x64 Updated	No Run	luonuo	71
4.6	---	luonuo	74
Windows 2008 R2	Not Covered	luonuo	76
12.1.2	---	luonuo	77
Mercury Application	---	luonuo	78
Mercury Tours Application	No Run	luonuo	79
2013	---	liyue	49
Windows 2012	Not Covered	liyue	50
1.系统登录	No Run	luonuo	80
RHEL 6U3 x64 Updated	No Run	luonuo	72

图 10-15　需求格的显示方式

需求模块包括如下的核心元素：

(1) Requirements Menu Bar，需求菜单栏，具有需求模块命令的下拉菜单。

(2) Requirements Toolbar，需求工具栏，具有创建或修改需求树的常用命令按钮。

(3) View，视图选择框，能选择需求树的显示方式：文档视图或覆盖视图。

(4) Requirements Tree，需求树，测试需求的一种图形表达。

(5) Description Tab，描述标签页，显示当前所选择需求的注释，仅在文档视图中有效。单击 Show 箭头显示描述面板。

(6) Attachment Tab，附件，为选中的需求提供附加信息，可存储文件、图片等内容。

(7) History Tab，历史标签页，显示当前所选择需求的历史操作列表。

(8) Tests Coverage Tab，测试覆盖标签页，显示了在需求树上，当前所选需求的测试列表。仅适用于覆盖视图。

(9) Details Tab，细节标签页，显示在需求树上当前树选择需求的详细描述。仅适用于覆盖视图。

3) Requirements Coverage

如图 10-16 所示。

4) Coverage Analysis

如图 10-17 所示。

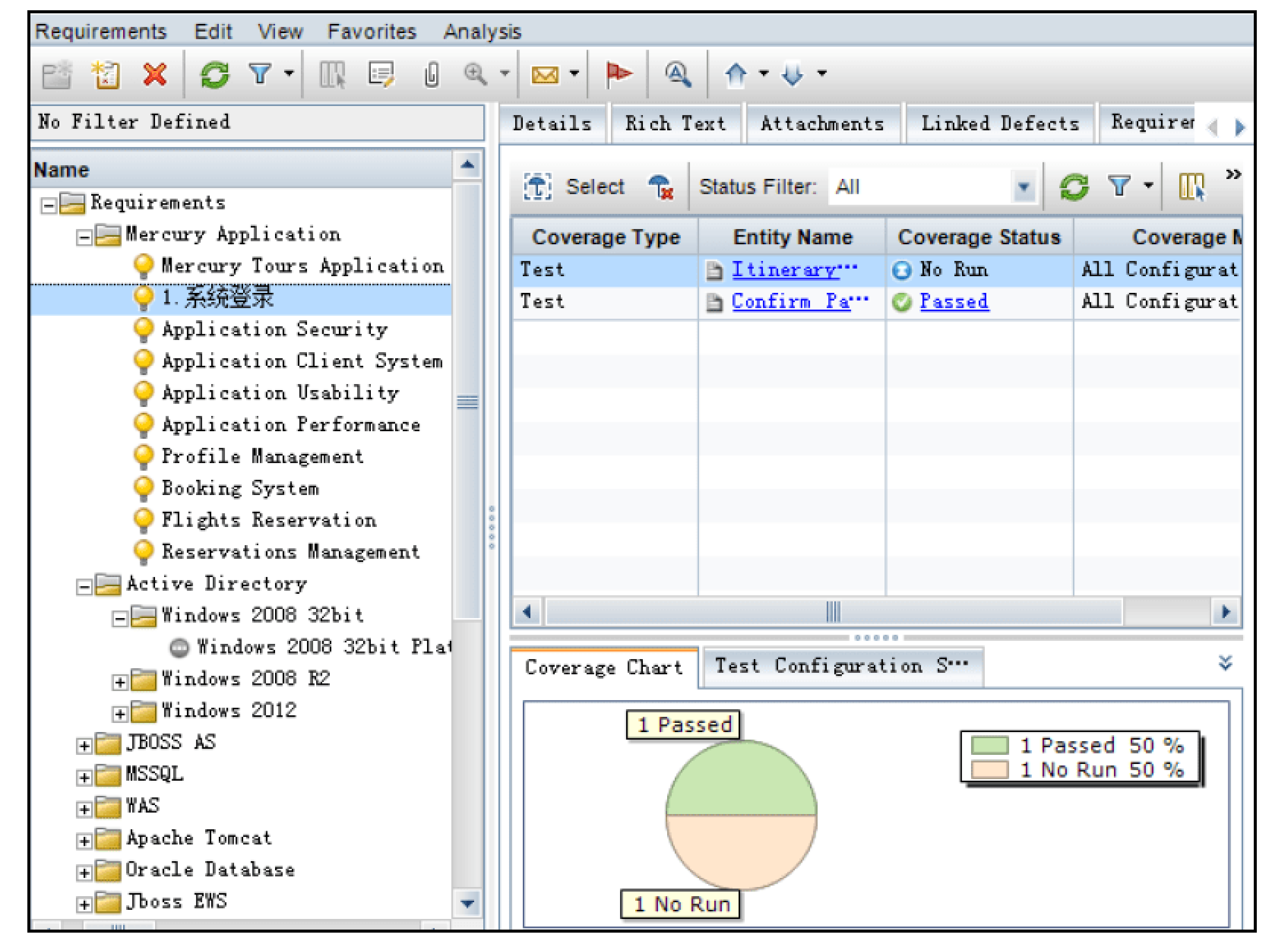


图 10-16　需求覆盖的显示方式

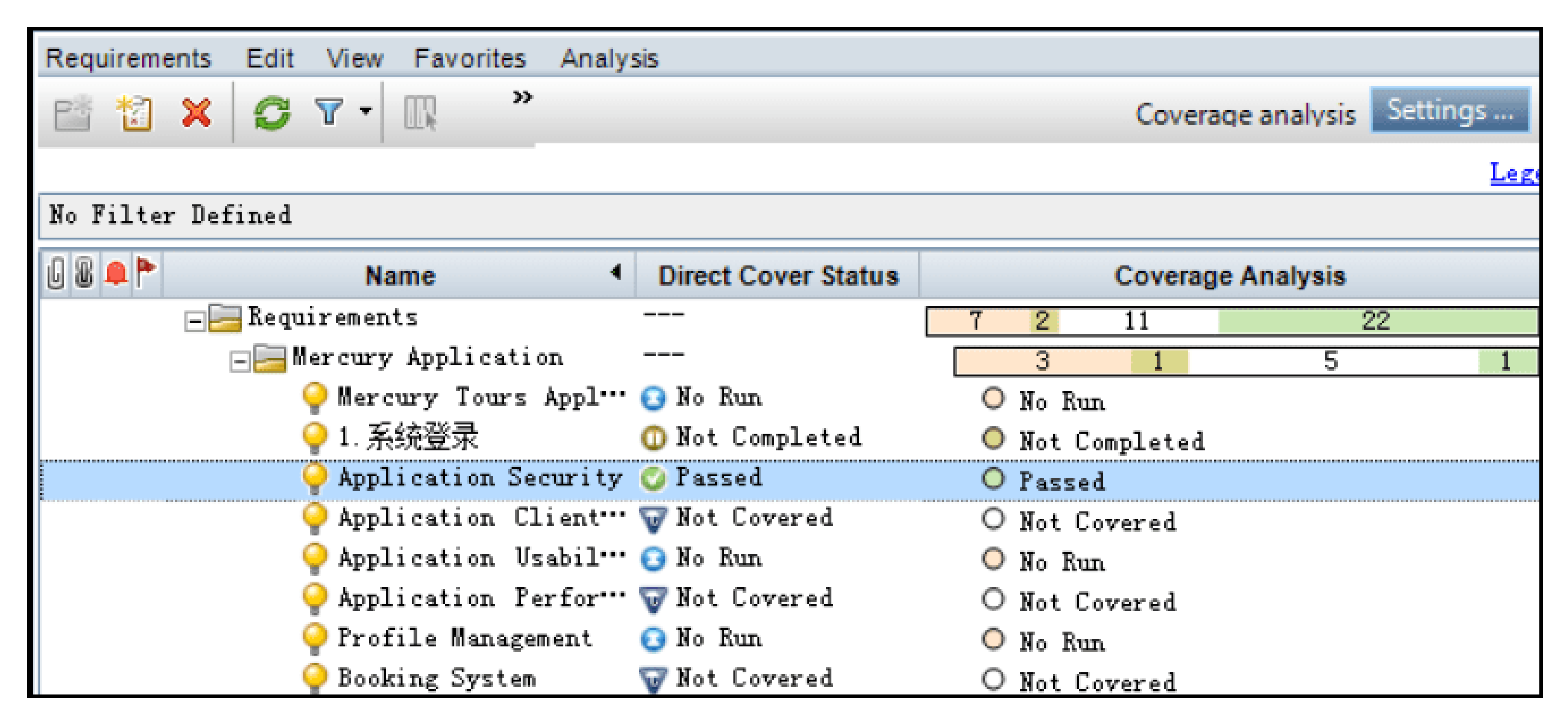


图 10-17　需求分析的显示方式

10.5.2　需求菜单栏

需求菜单栏包括如图 10-18 所示的菜单。



图 10-18　需求菜单栏界面

1) Requirements 菜单，包括命令：在需求树上修改需求、从一个需求产生一个测试、邮寄一个需求。

2) Edit 菜单包括命令剪切、复制、粘贴、重命名,以及删除需求和查找、替换搜索需求。

3) View 菜单，包括命令：设置需求树的显示、查找一个需求、浏览测试覆盖、关联缺

陷和附件。

4) Favorites 菜单，包括增加、组织个人喜欢使用的命令。

5) Analysis 菜单，包括生成需求报告和图表的命令。

10.6　可跟踪矩阵概述

可跟踪性矩阵允许你确定需求和需求之间以及需求和测试之间的关系的范围。它有助于你验证是否满足所有需求，如有更改还可标识更改的需求范围。

可跟踪性矩阵列出源需求及其关联的需求和测试。对于每个源需求都会列出关系总数。低值可能表示源需求关联的需求或测试不够。高值可能表示源需求太复杂，可以进行简化。零值表示不存在关联关系。

10.7　覆盖率分析

可在测试计划模块的需求覆盖标签页中找到你准备打开的需求，并可以直接在需求模块中将其打开。同样可在需求模块的测试覆盖标签页中找到你准备打开的测试，并直接在测试计划模块或测试实验室模块中将其打开。在 ALM_Demo 项目中，选择需求 Mercury Tours Application 下面的需求 Application Client System。此需求有 12 个子项(包括自身)。在 Coverage Analysis 中，可看到两个子项的状态为失败(一个或多个由此需求覆盖的测试失败)。进行分析时，可看到与此需求关联的三个(27%)测试失败。如图 10-19 所示。

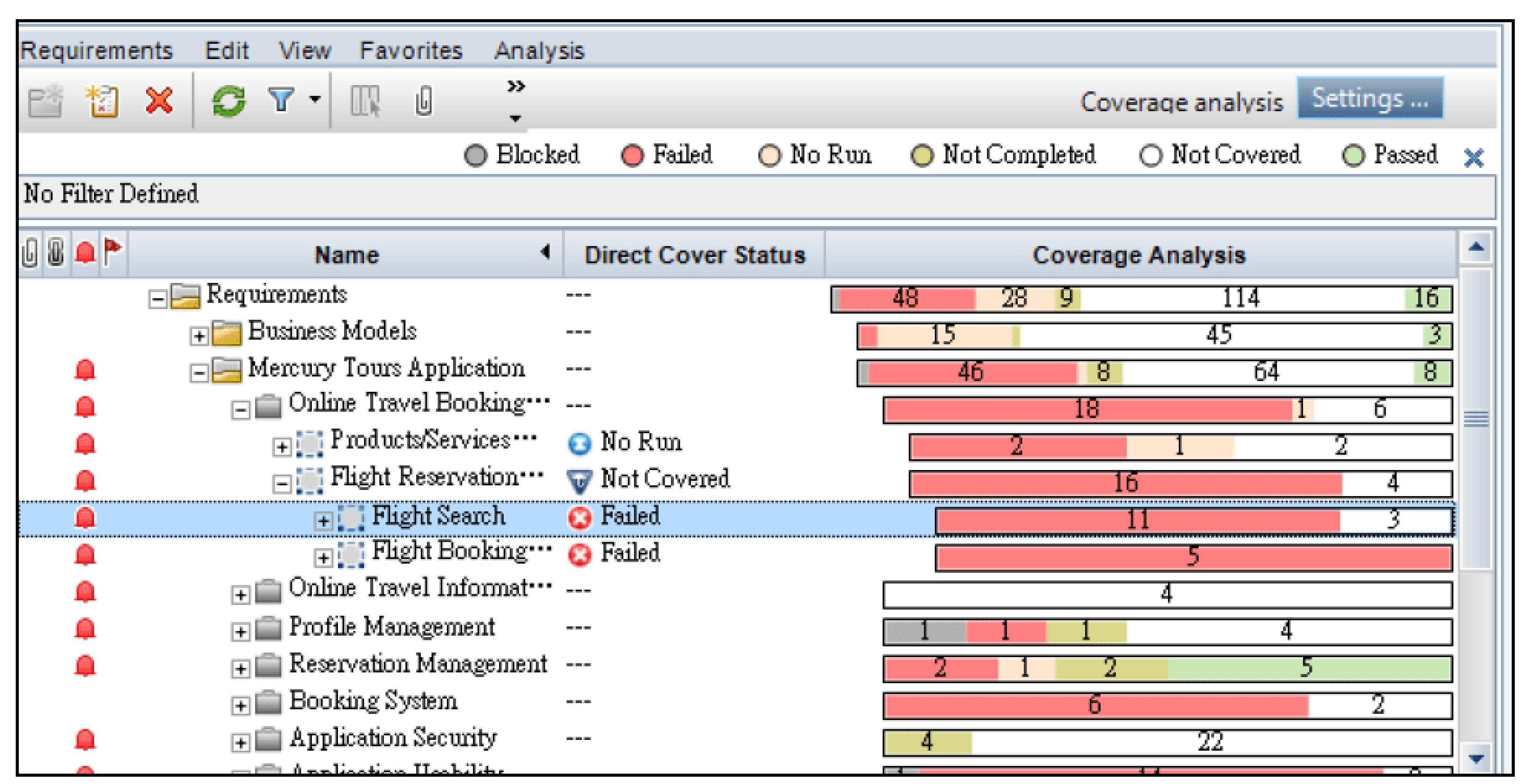


图 10-19　覆盖率分析

习题与思考题

1. 如何观察覆盖分析中的说明？
2. 如何理解需求分析？
3. 如何使用测试配置生成需求覆盖范围？

练习：创建需求树

在本次练习中，需完成以下任务：

第 1 部分：创建需求树
第 2 部分：创建需求
第 3 部分：填写 Details 面板需求信息
第 4 部分：查看需求不同状态
(该练习的指导步骤请参见本章正文)

第11章　测试计划

11.1　测试计划概述

应用生命周期管理路线图如图 11-1 所示。应用生命周期管理路线图测试流程的第三步是测试计划，在需求经过批准后，测试团队会设计测试来检验程序是否满足这些需求。测试的效率取决于你如何计划和执行这些测试。

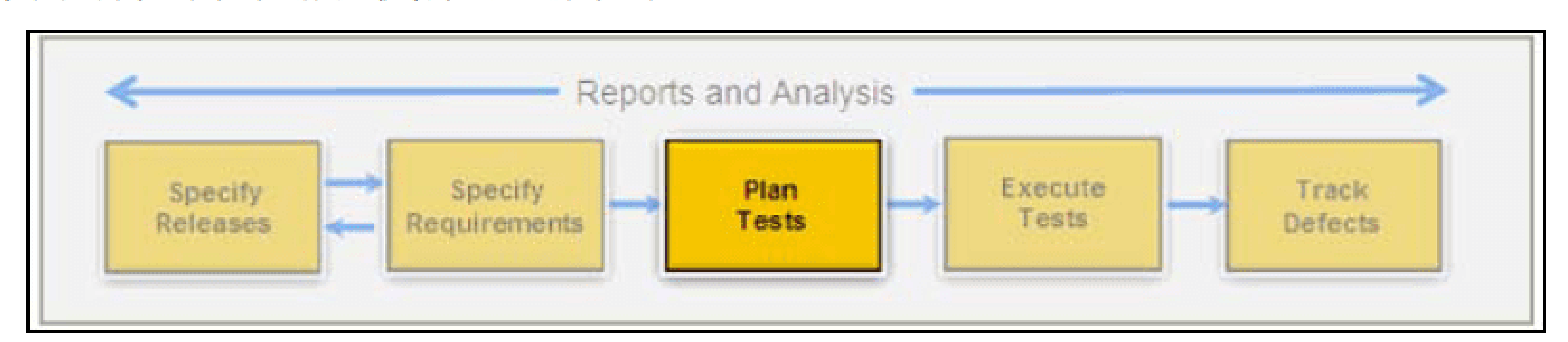


图 11-1　测试计划

通常的应用程序都太大，无法将其作为一个整体进行测试。测试计划模块可以根据功能来划分应用程序。通过创建 Test Plan Tree，可将应用程序分为若干单元或者 Subject。测试计划树是测试计划的图形表示，可以按照测试功能的层次关系来显示测试。在树中定义主题后，可以为每个主题创建测试，然后将其添加到树中。在这个阶段，你定义树的一些基本信息，例如树的名字、状态和设计者。你还可以附加一些文件、URL 程序快照或说明测试的系统信息。然后你定义测试的步骤，其中包括：如何运行测试和期望结果的详细说明，你需要开发一个清晰、简明的测试计划来成功的测试一个程序。一个好的测试计划可在测试流程的任何阶段对你的程序质量进行评价。

11.1.1　如何计划测试

1) 先决条件

需求树中定义的一系列需求，你可以直接根据需求模块中的需求来自动创建测试。

2) 创建测试计划树

新建测试主题文件夹和新建测试来创建测试计划树。

3) 创建测试资源依赖关系——可选

你可以将测试和上传到 ALM 存储库中的——系列资源联系到一起，然后查看这些依赖性和确认哪些资源正在使用。

4) 定义测试参数

为了让测试更加灵活，在测试过程中包含一些参数。可以反复运行同一个测试程序并对参数指定不同的值。

5) 定义测试配置

为了在不同的使用案例中运行测试，你可以定义测试配置。这可以让你在不同的场景下运行同一个测试程序。

6) 创建测试步骤

创建测试步骤描述执行操作和他们期望的结果。定义完测试步骤后，需要决定是采用手动测试还是自动测试。

7) 自动化测试

设计测试步骤后，需要决定哪些采用自动化测试。影响自动测试的因素包括执行的频率、输入的数据量、执行测试的时间长度和复杂性。可以创建一个自动化系统测试来提供一台机器的系统信息，捕获桌面图像或重启机器。

8) 测试计划树中的每个测试都连接需求树中的一个或多个需求

通过定义测试的需求范围。你可以确保测试计划中的测试与原始的需求保持一致

9) 将缺陷和测试相关联

将测试与特定缺陷相关联。这十分有用，例如在专门针对已知缺陷新建测试时。通过创建关联，可以确定是否应该基于缺陷的状态运行测试。

10) 分析测试计划的数据

通过生成图表和报告来分析测试计划。使用下面的一种：

(1) 显示测试主题的动态图表。在测试计划树中，选择一个测试主题，然后单击活动分析在图表中展示测试计划数据。

(2) 在测试计划模块菜单中，选择 Analysis|Graph 创建测试计划数据的报表。

(3) 在测试计划模块菜单，选择 Analysis|Report 来创建报表。

11) 设立基线

当测试计划被检查和批准后，可设定一个基线。基线可以为你提供测试计划在特定时间点的一个快照。你可以使用基线来标记应用周期中的重大事件。此外基线还可以作为一个比较差异的参考点。

11.1.2　开发测试计划

1) 开发测试计划树；
2) 在测试计划树创建主题文件夹；
3) 在每个主题文件夹中定义指定的测试程序；
4) 为每个测试添加步骤说明；

5）视情况建立测试脚本；

6）将测试和需求相关联；

11.2　测试计划模块

如图 11-2，所有测试计划任务都是从测试计划模块执行的。为了熟悉使用这些模块，单击 ALM 侧边栏的 Testing 组下面的 Test Plan 图标。

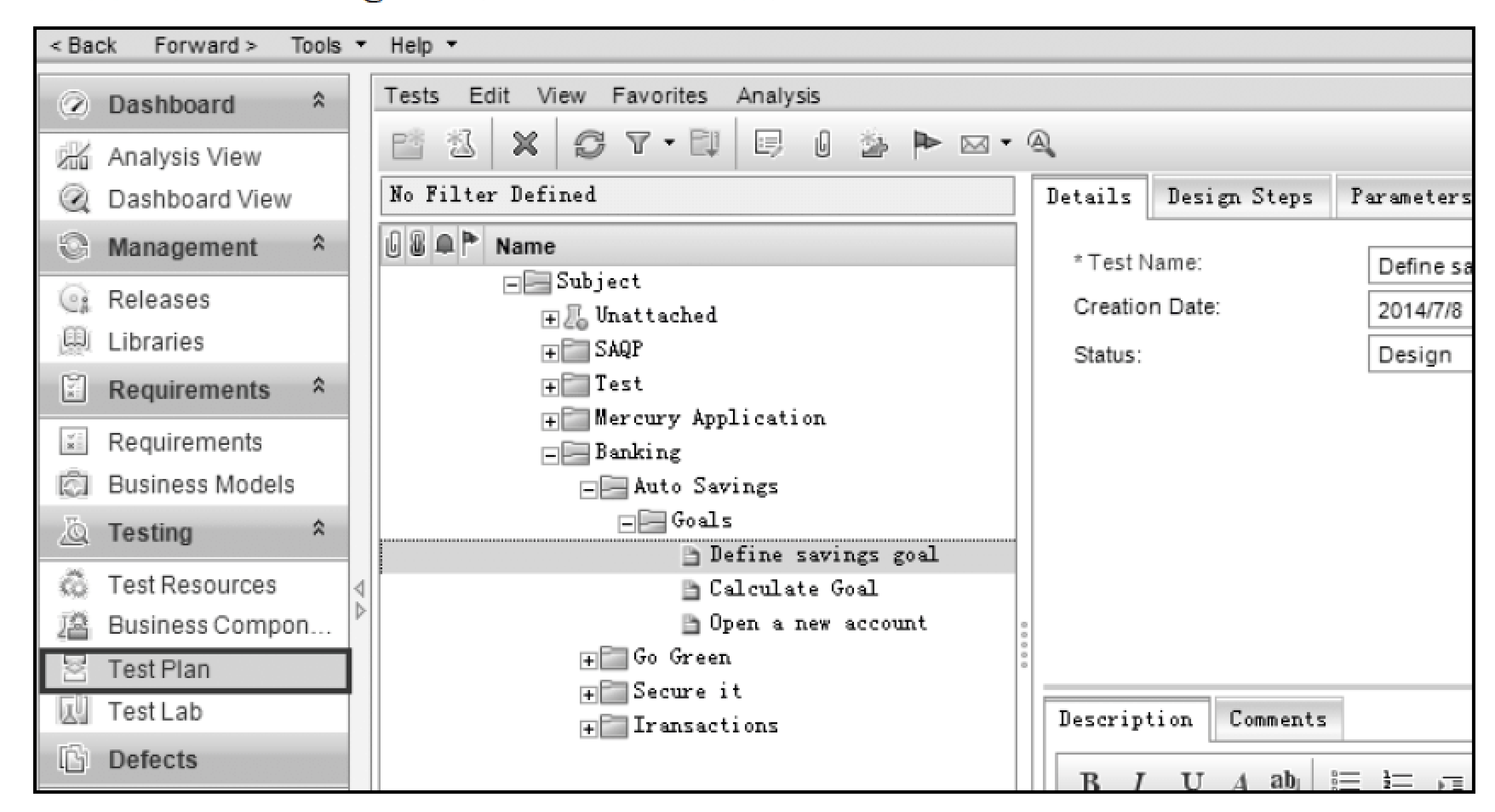


图 11-2　测试计划模块

11.2.1　测试计划树

根据功能单元和程序主题来整理测试为测试构建模块提供一个清楚的图片，以此来帮助开发真实的测试，显示测试之间的阶层关系和依赖性。

在测试计划模块的左边窗口显示的是测试计划树。测试计划树是项目的测试计划的图形表示。它包含在根节点下提交文件夹。在主题文件夹下创建文件夹。

在计划测试计划树时，需要考虑程序中功能间的层级关系，将这些功能划分成主题来建立一个测试计划树，用这个树来代表程序的功能。图 11-3 的例子显示的是银行业务和航班预定程序的测试计划树。银行业务下的主要测试主题有 Auto Savings、Go Green、Inquiry、Secure It、Transactions，这些作为第一层文件夹出现。为了分解那些复合的主题，Auto Saving 下有第

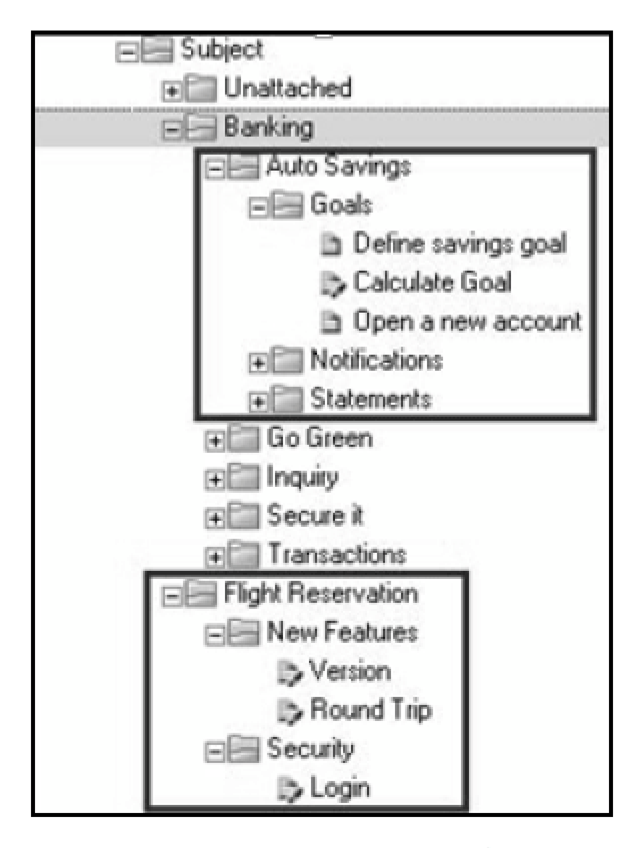


图 11-3　银行业务和航班预定程序的测试计划树

二层文件夹或主题。

11.2.2　创建测试计划树

可以采用两种方式来创建测试计划树：

1) 将需求转换为测试：你可以根据需求模块下的需求来自动创建测试。

2) 手动创建测试计划树：包括手动创建主题文件夹和将测试添加到这些文件夹。

11.3　连接测试到需求

对整个测试来说，将测试计划树中的测试与原始的测试需求对应起来是基础。可通过连接测试计划树中的测试到需求树中的一个或多个需求来创建测试覆盖。本节描述了以下内容：

1) 连接测试到需求(Linking Tests to a Requirement)

2) 连接需求到测试(Linking Requirements to a Test)

3) 连接需求和测试覆盖(Linking Requirements and Tests Coverage)

11.3.1　连接测试到需求

对于整个测试过程，首先在需求树上定义测试需求。在测试计划阶段，再基于这些测试需求构造测试计划树。为在需求和测试之间建立关系，应在 ALM 中增加它们之间的连接。一旦测试也连接到缺陷，就能确定整个测试过程对测试需求的依从情况。如果一个测试需求改变，能够立即确定是哪些测试和缺陷受到影响以及具体的影响。

在测试计划模块，可通过选择需求连接到一个测试来创建需求覆盖。也可以在需求模块，通过选择测试连接到一个需求来创建测试覆盖。一个测试能覆盖一个或多个需求，一个需求也可以覆盖一个或多个测试，如图 11-4 所示。

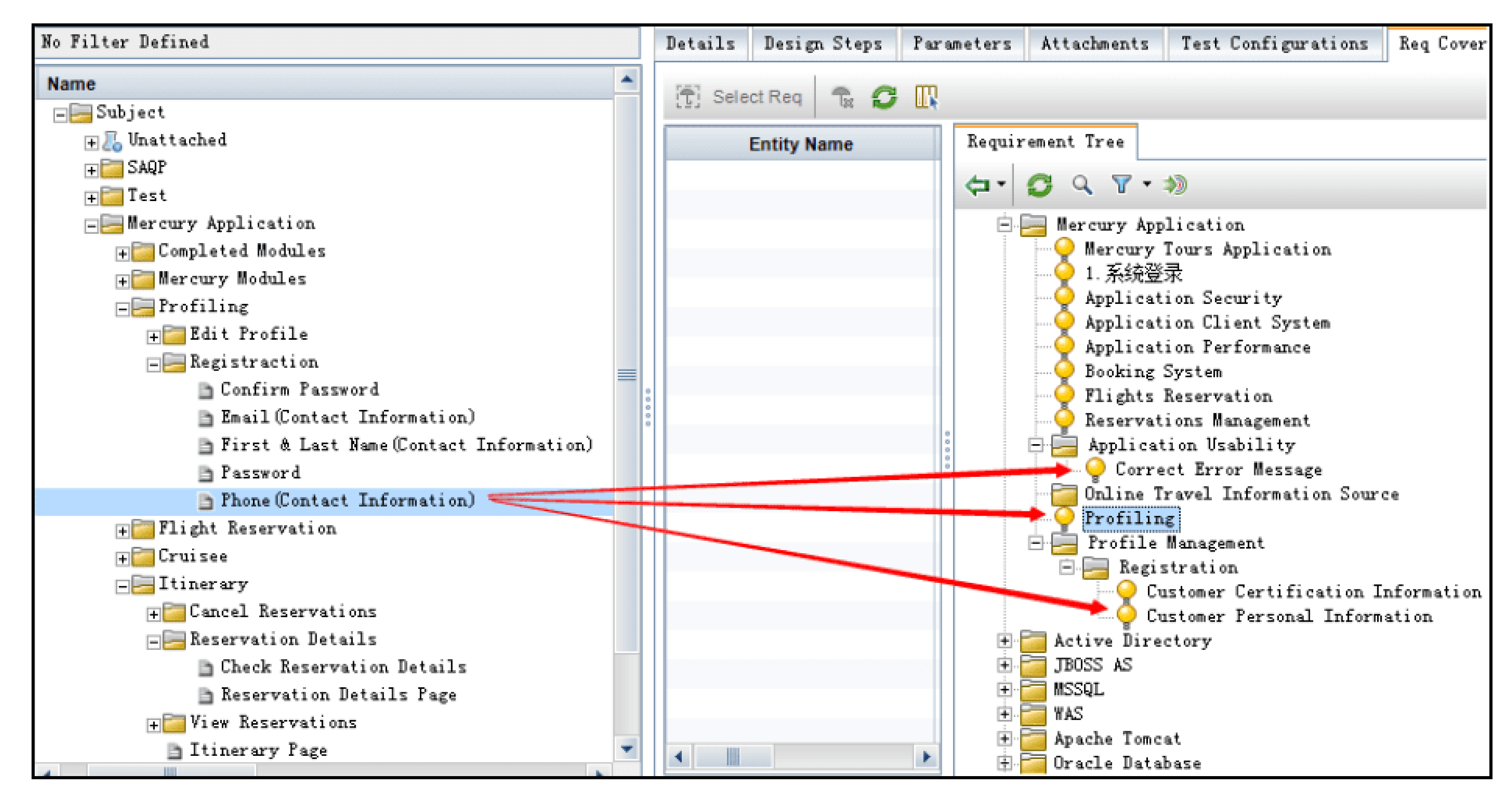


图 11-4　连接测试到需求

例如，对于工程中的航班预订系统。单击 Test Plan 标签页，在测试计划树的 Profiling 文件夹下，展开 Registration 文件夹并选择 Phone(Contact Information)测试。

Phone(Contact Information)测试是检查用户的联系电话是否为空。假如你单击 Requirements Coverage 标签页，能够看到该测试覆盖了如下的需求主题：Mercury Tours Application、Profiling、Application Usability、Correct Error Messages、Profile Management、Registration 和 Customer Personal Information。

也可在工程中，以另外的视角来查看覆盖。单击 Requirements 标签页，在需求树的 Profile Management 文件夹下，展开需求主题 Registration 并选择 Customer Personal Information，如图 11-5 所示。

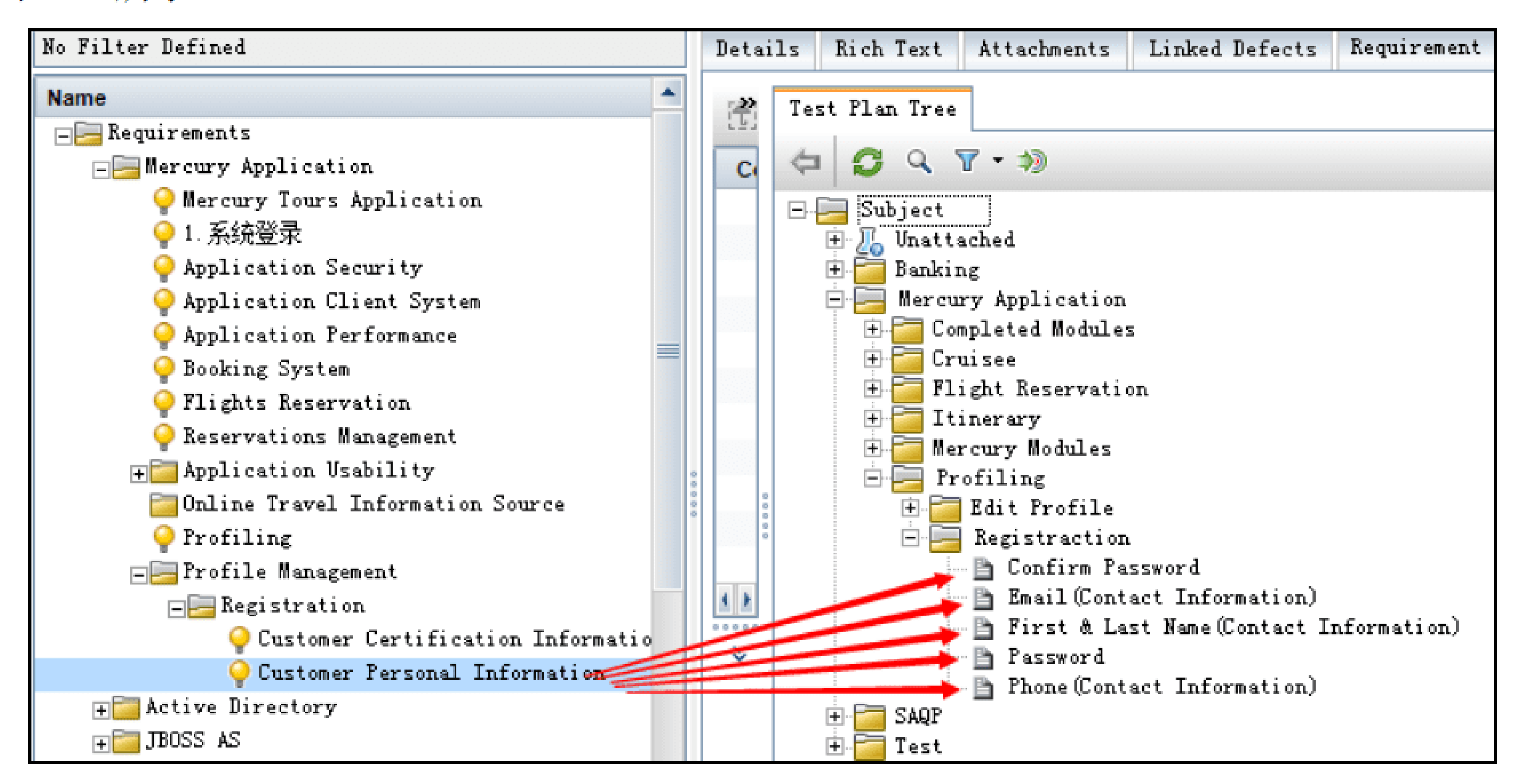


图 11-5　查看测试对需求的覆盖的另一种方法

Customer Personal Information 需求是航班预订系统所含的客户个人信息。在 Tests Coverage 标签页，也可看到此需求被如下的测试覆盖：Registration、First & Last Name(Contact Information)、Phone(Contact Information)、Email(Contact Information)和 Mailing Information。

1. 查看测试覆盖网格

能够过滤覆盖网格，显示或隐藏所有覆盖。从 Status Filter 列表中选择一个状态，通过状态类型过滤覆盖网格。例如，可选择仅去查看所有已经通过的测试。选择 All 清除过滤。选中 Full Coverage 复选框，将显示所有测试覆盖，包括子需求的测试覆盖。默认情况下，Full Coverage 是没有被选中的。

2. 添加测试覆盖

可从测试计划树上选择一个或多个测试，来添加测试覆盖到一个需求。

1) 在需求树上选择一个需求。测试覆盖标签页显示了所选择需求的覆盖网格。

2) 在测试覆盖标签页，单击 Select Tests 按钮来显示测试计划树(在右边，如图 11-6 所示)。

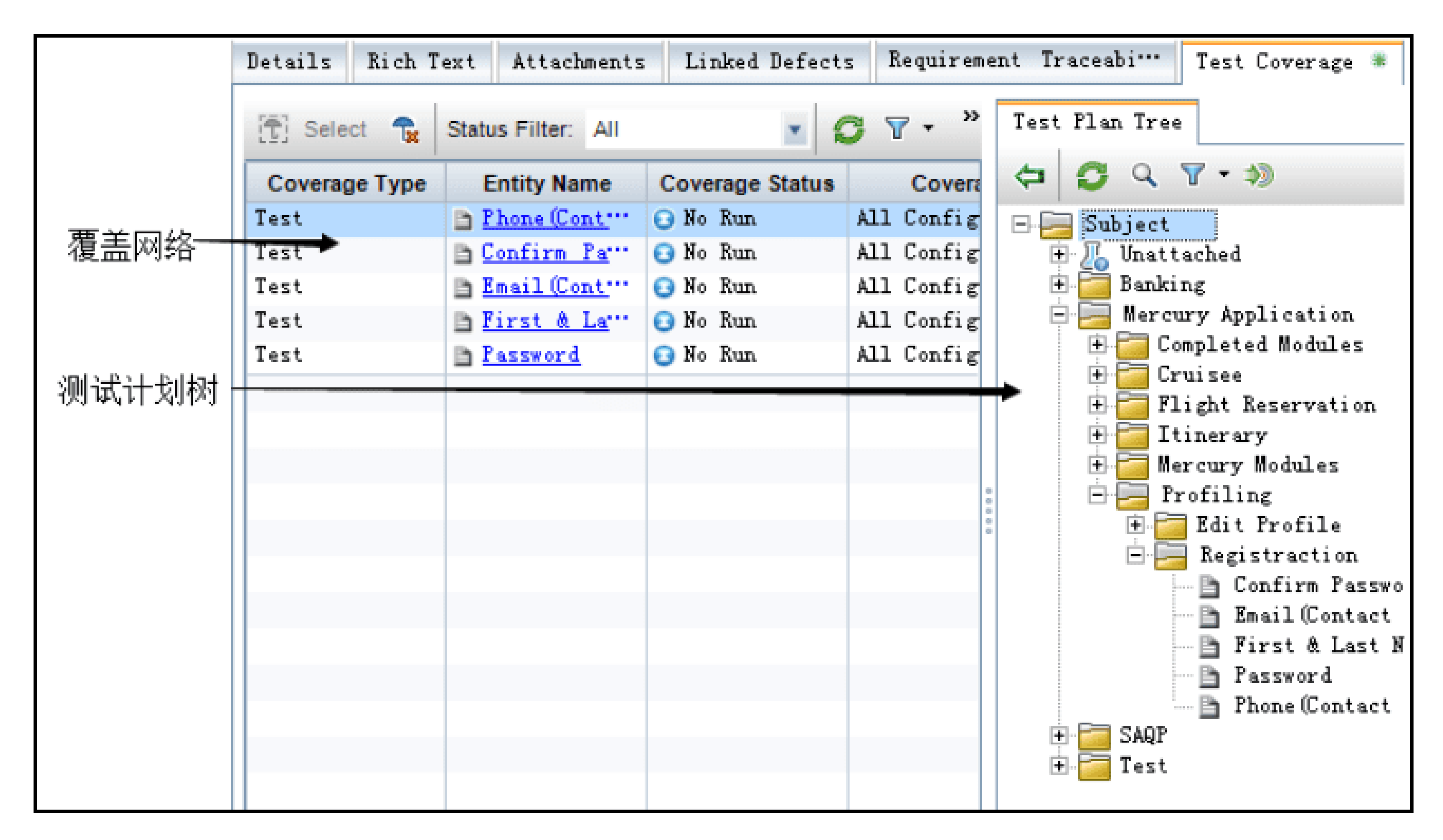


图 11-6　测试计划树显示

3) 在树中搜索一个指定测试：在 Find 输入框中输入所要搜索的测试的名称(或部分名称)，并单击 Find 按钮。假如搜索成功，ALM 会在树中高亮显示此测试。

4) 在树中刷新一个测试：选择测试，并单击 Refresh Selected 按钮。若想刷新树中所有的测试，选择 Subject 文件夹，并单击 Refresh Selected。

5) 选择一个测试或测试文件夹，单击 Add to Coverage 按钮，为所选需求添加测试覆盖。测试被添加到测试覆盖网格中。

6) 单击 Close 按钮来隐藏测试计划树。

提示：可通过将测试计划树中的测试或测试文件夹拖动到覆盖网格中，来定义测试覆盖。

3. 移除需求覆盖

可从一个需求的测试覆盖中删除一个或多个测试。在需求树上选择一个需求。测试覆盖标签页显示所选需求的覆盖网格。

1) 从覆盖网格中选择一个或多个准备删除的测试。

2) 单击 Remove from Coverage 按钮，并单击 Yes 确认。

11.3.2　连接需求到测试

在测试计划期间，在测试计划树上选择一个测试时，ALM 会在需求覆盖标签页中显示这个测试的需求覆盖。覆盖网格中列出了所选测试覆盖的需求。可在这个覆盖网格中添加或删除需求，如图 11-7 所示。

提示：右击覆盖网格，并选择 Show Full Path，可显示需求在需求树中位置。需求网格包含如下的列，如表 11-1 所示。

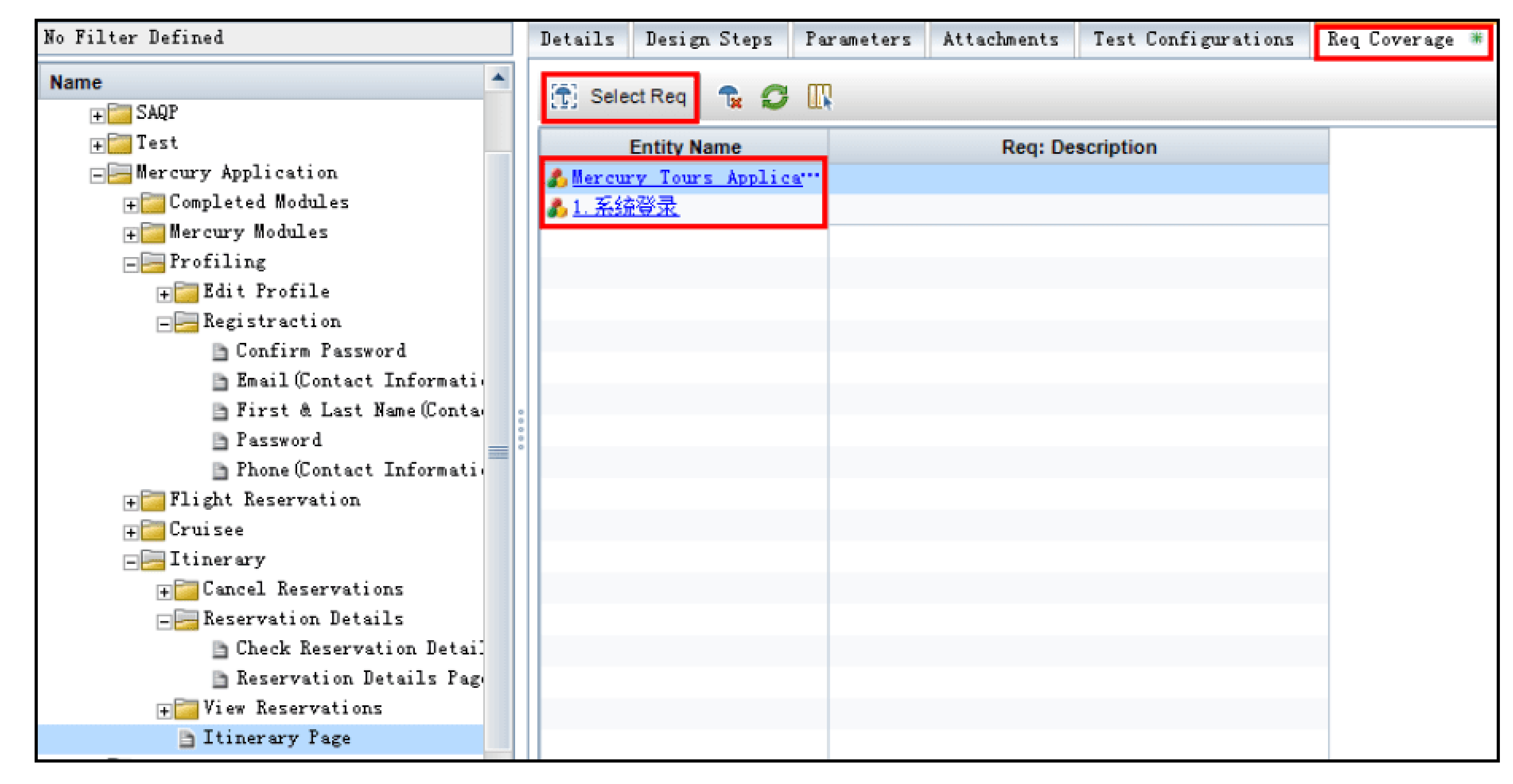


图 11-7　在覆盖网格中添加或删除需求

表 11-1　需求网格列表内容

列	描　述
Coverage Type	覆盖类型
Entity Name	实体名
Entity Status	实体当前状态
Req：Author	需求创建人
Req：Creation Date	需求创建日期
Req：Creation Time	需求创建时间
Req：Description	需求描述
Req：direct cover status	需求当前状态。默认状态为 Not Covered Not Covered：需求没有关联到测试。 Failed：一个或更多被需求覆盖的测试执行状态为 Failed Not Completed：一个或更多被需求覆盖的测试执行状态为 Not Completed Passed：所有被需求覆盖的测试执行状态为 Passed No Run：所有被需求覆盖的测试执行状态为 No Run N/A：不适用
Req：Modified	需求最后改变的时间
Req：name	需求名称
Req：priorty	需求优先级
Req:Product	需求基于的应用程序组件
Req:Reviewed	需求是否经过审核
Req:Type	需求类型，可以是硬件或者软件

1. 添加需求覆盖

1) 从需求树上选择需求，将需求覆盖添加到一个测试。

2) 在测试计划树上选择一个测试。

3) 单击 Req Coverage 标签页。

4) 单击 Select Req 按钮，在右边显示需求树(如图 11-8 所示)。

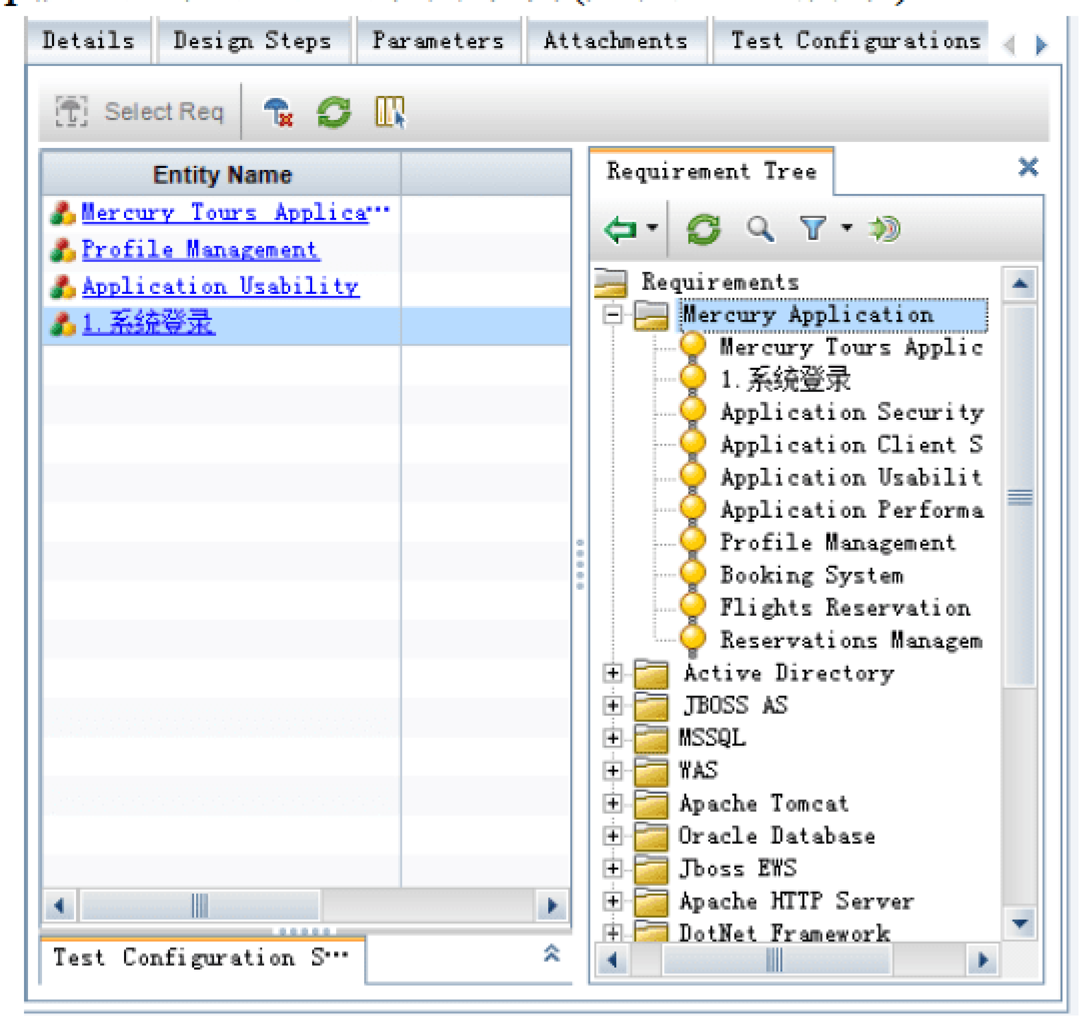


图 11-8　需求树显示

5) 在需求树中搜索特定的需求：在 Find 输入框中输入所要搜索的需求的名称(或部分名称)，并单击 Find 按钮。如果搜索成功，ALM 会在树中高亮显示此需求。

6) 在树中刷新一个需求：选择需求，并单击 Refresh Selected 按钮。若想刷新树中所有的需求，右击此需求树，并选择 Refresh | Refresh All。

7) 选择一个需求或需求主题添加覆盖。

8) 如果你想覆盖能够包括所选需求的子需求，选择 Include Child Requirements into Test Coverage。

9) 单击 Add to Coverage 按钮，需求被添加到覆盖网格中。

提示： 可通过将需求树中的需求或需求主题拖动到覆盖网格中，来定义需求覆盖。单击 Close 按钮隐藏需求树。

2. 移除需求覆盖

你可以从测试的需求覆盖中删除一个或多个需求。

1) 在测试计划树上选择一个测试。

2) 单击 Req Coverage 标签页。

3) 从覆盖网格中选择一个或多个准备删除的需求。

4) 单击 Remove Selected 按钮，并单击 Yes 进行确认。

11.3.3　分析覆盖

可在测试计划模块的需求覆盖标签页中找到准备打开的需求，并可以直接在需求模块中将其打开。同样也可在需求模块的测试覆盖标签页中找到准备打开的测试，并可以直接在测试计划模块或测试实验室模块中将其打开。

1. 连接到需求树上的需求

1) 在测试计划树上选择一个测试，并单击 Req Coverage 标签页。

2) 在 Requirement 列，单击你想在需求树上查看的需求。或者右键单击需求，并选择 Find in Requirements Tree。

2. 连接到测试计划树或测试集上的测试

在需求树上选择一个需求。测试覆盖标签页显示显示了所选需求的覆盖网格。

1) 在测试计划树上打开一个测试：在覆盖网格上右键单击此测试，并选择 Go to Test in Test Plan Tree。

2) 在测试实验室模块上打开一个测试：在覆盖网格上右键单击此测试，并选择 Go to Test in Test Set。

11.4　将需求转化为测试

创建需求树后，需求将作为基础用来在测试计划模块中定义测试计划。ALM 中有固定的元件来将你的项目需求转换为测试。

在将需求转换为测试时会自动创建需求和测试间的一对一匹配。当需求转换为测试时，需求树中的阶层关系也会复制到测试计划树中。另外，在转换过程中，ALM 可以让你决定是否将个别的需求转换为一个文件夹、测试或一个设计步骤。

如果你未指定测试过程中不满足需求的影响，将需求转换为测试之后，你需要手动指定测试过程的不满足需求的影响。而且，还需要为每个测试都指定通过和不通过的条件。例如，一个登录过程的程序需求将定义用户名和密码字段的有效值，这些需求能直接转换为测试，并根据每个字段允许的值来简单决定测试通过或失败的条件。另外，任何一个测试的失败都意味着程序使用者并不被认可，这样的结果对测试过程有严重影响。

11.4.1　选择自动方法来转换

将需求转换为测试时，需要选择一个自动转换的方法并决定是否将需求转换为设计步骤、测试或测试文件夹，如图 11-9 所示。

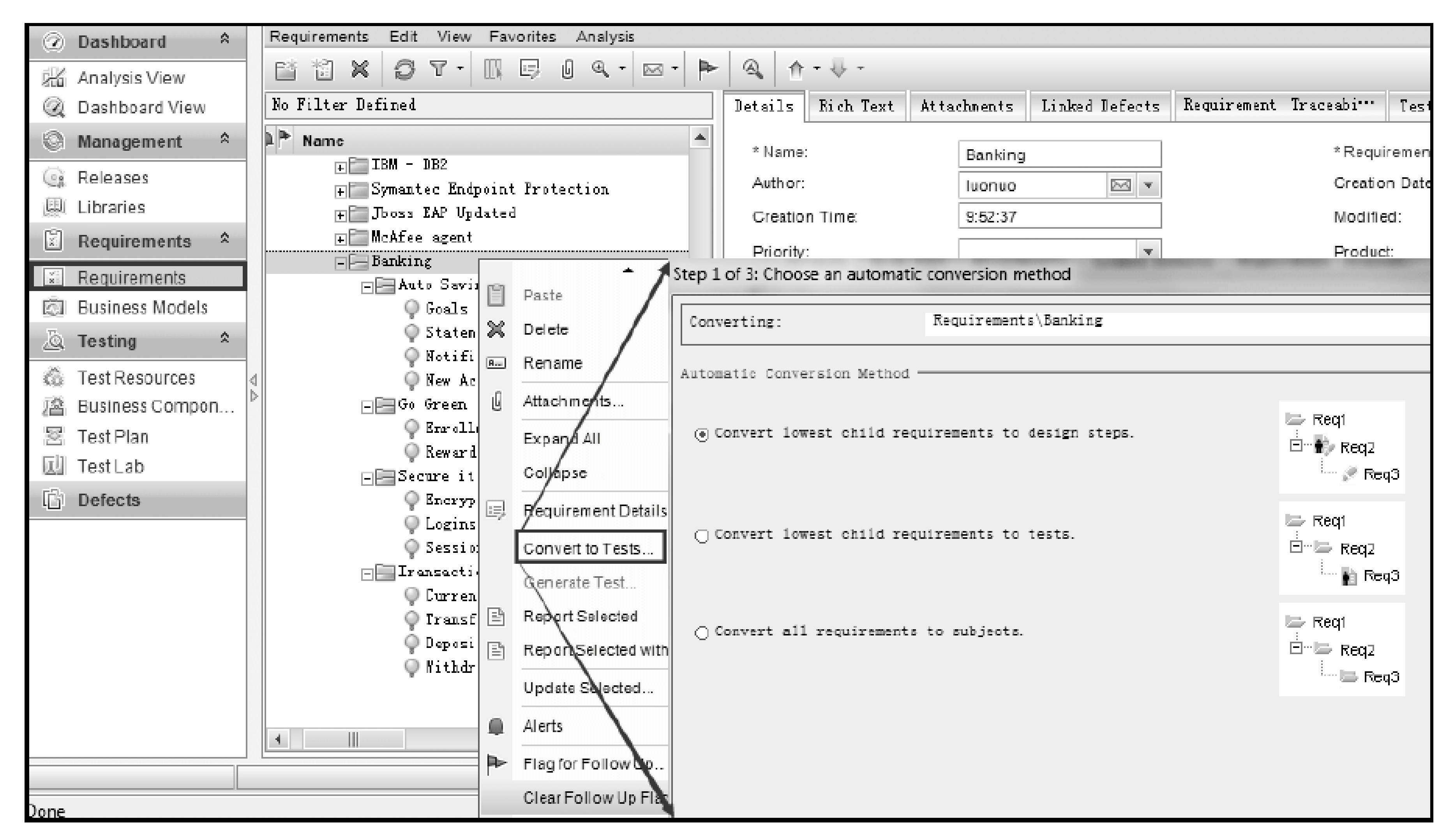


图 11-9　自动化方法选择

为转换选择自动方法的步骤如下。

1) 在 ALM 的工具条，单击 Requirements 组下面的 Requirements 图标。

2) 从 ALM 的菜单栏中，选中 View | Requirements Tree。

3) 从 Requirements 树中，选中一个需求。

4) 从 ALM 菜单栏中，选中 Requirements | Convert To Tests，弹出一个 Choose an Automatic Conversion Method 的对话框。

5) 在选中自动转换对话框中，在自动转换方法下面，选中一个转换方法然后单击 Next，将弹出一个 Manual Changes to the Automatic Conversion 对话框。

11.4.2　改变自动化转换

在选择完一个自动转换方法后，可以自定义转换。例如，可选择 Convert lowest child requirements to design steps 这个自动转换方法。在转换过程中，你可以看到一些转换的需求在测试计划树中变成了文件夹。ALM 可以让你选中一个需求并将其转换为一个测试计划文件夹。如图 11-10 所示。

对自动转换进行变更：

1) 在 Manual changes to the automatic conversion 对话框中，从 Name 列表中选中一个需求。

2) 在工具栏中，单击下面一个按钮。

- Convert to Subject：将选中的需求转换为主题文件夹。
- Convert to Test：将选中的需求转换为测试。
- Convert to Step：将选择的需求转换为测试步骤。
- Convert to Description：将选择的需求转换为测试的描述。
- Exclude from Conversion：不转换选中的需求。

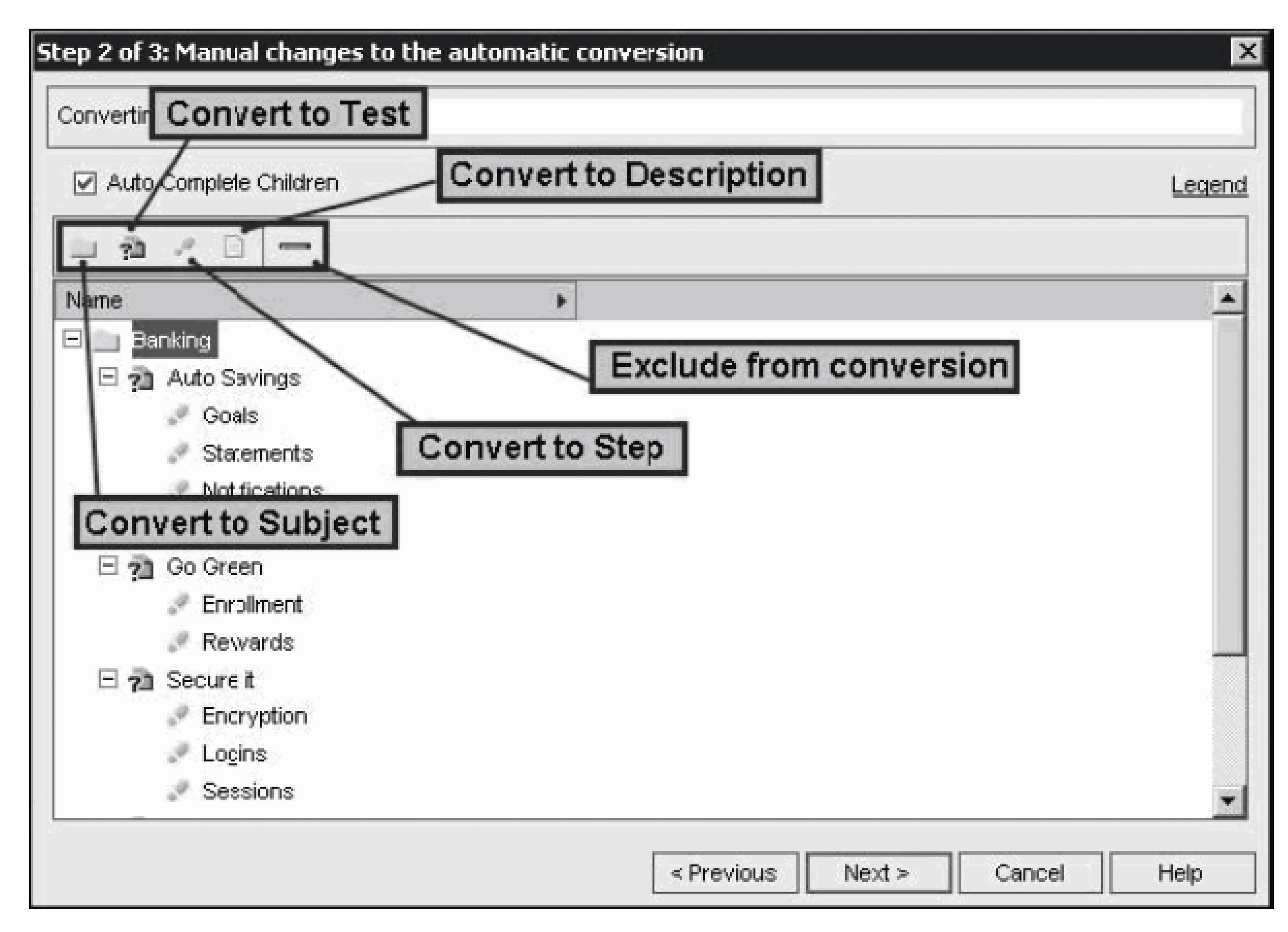


图 11-10　手动修改自动化测试转换

3) 单击 NEXT，弹出 Choose the destination subject path 对话框。

在手动更改自动测试转换后，需要为新建的测试选择目标路径。见图 11-11。

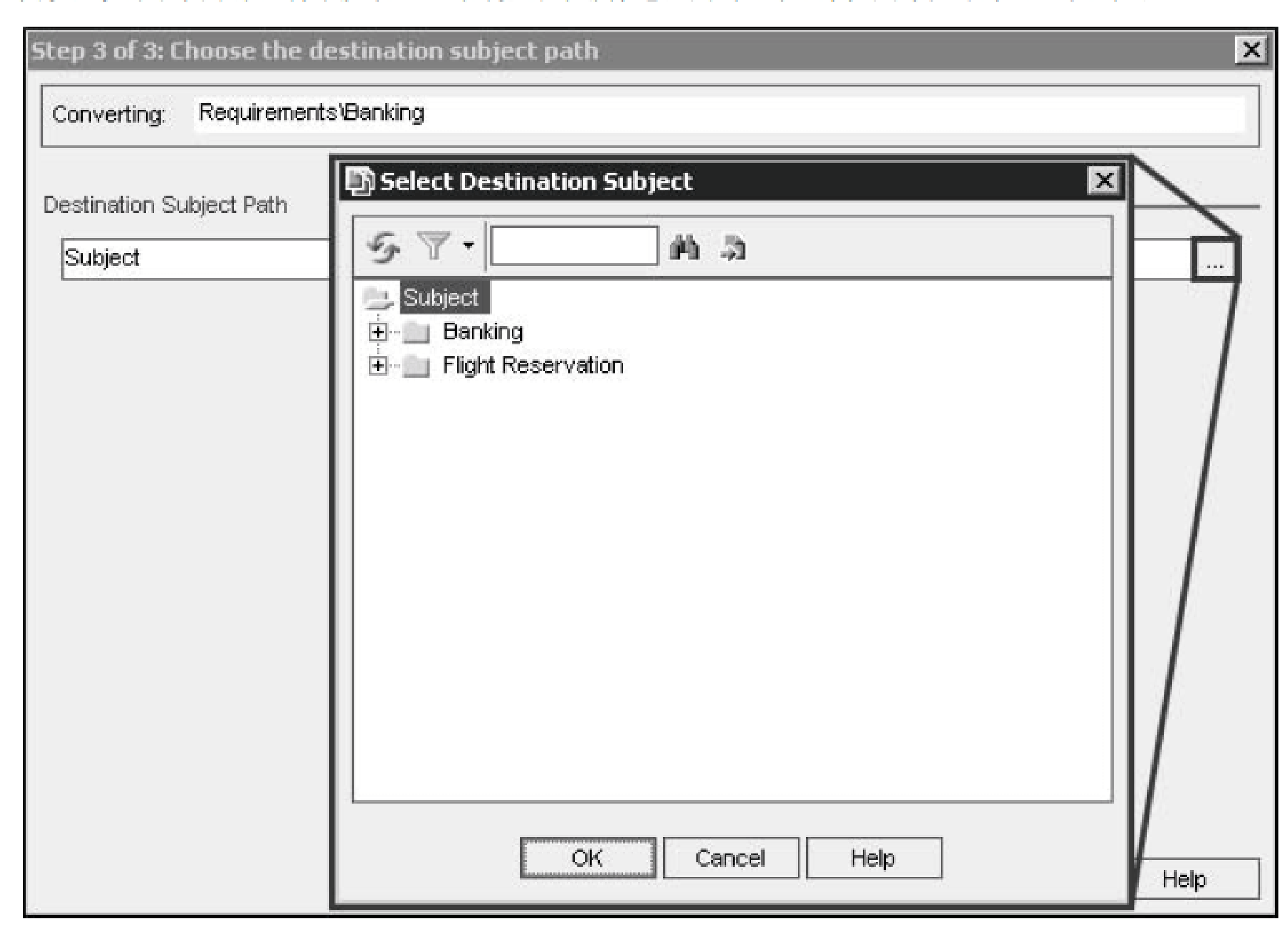


图 11-11　目标路径选择

选择目标路径：

1) 在 Choose the destination subject path 对话框中，单击 Destination Subject Path 文本框中的 View 按钮。弹出 Select Destination Subject 对话框。

2) 在 Select Destination Subject 对话框，从测试计划树中选择测试计划文件夹，然后单击 OK。

3) 单击 Finish 来关闭 Choose the destination subject path 对话框。Information 消息框会通知你转换过程成功完成。

4) 单击 OK 来关闭 Information 消息框。

11.4.3　创建主题文件夹

图 11-12 和图 11-13 对话框可让你将测试计划树下的文件夹归类，还可以根据你的习惯来定制类别。

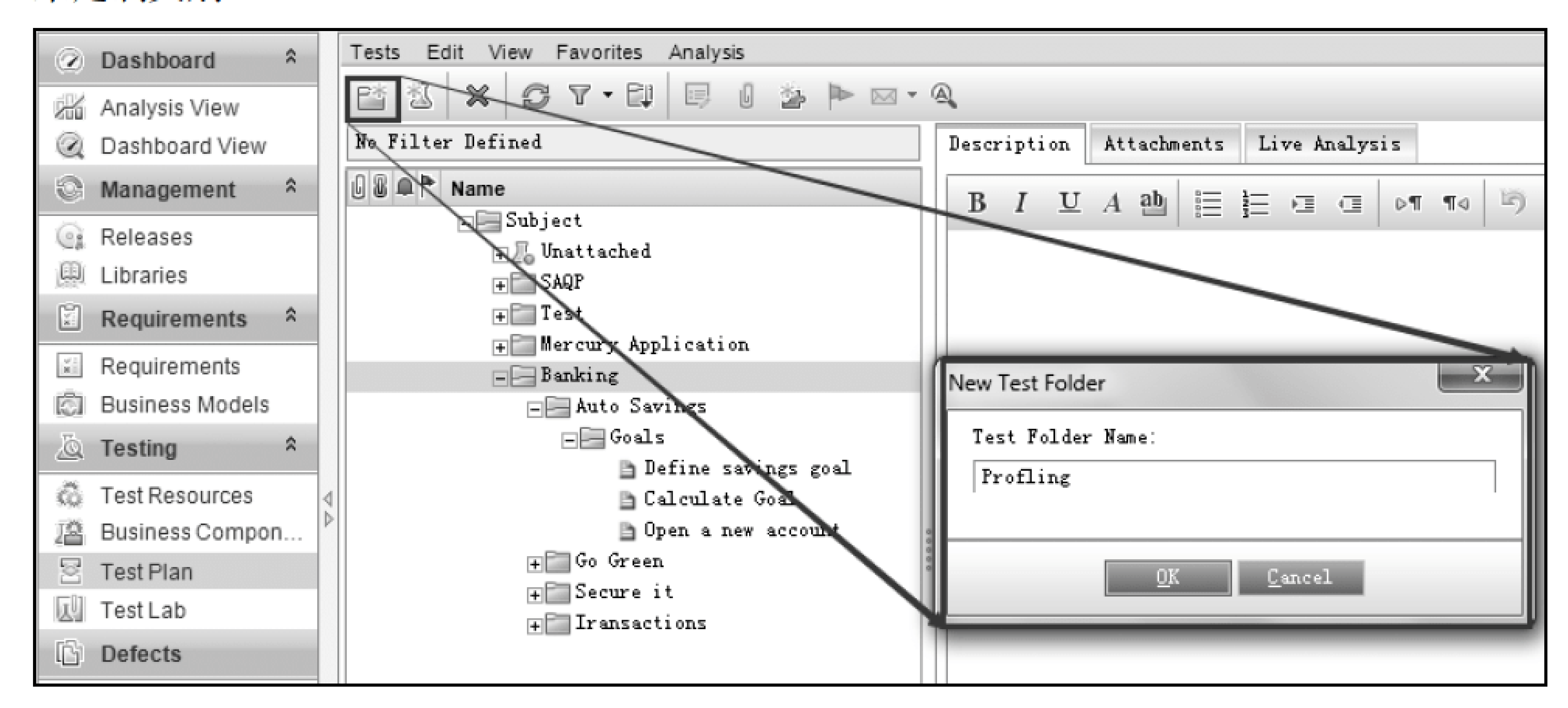


图 11-12　主题文件夹

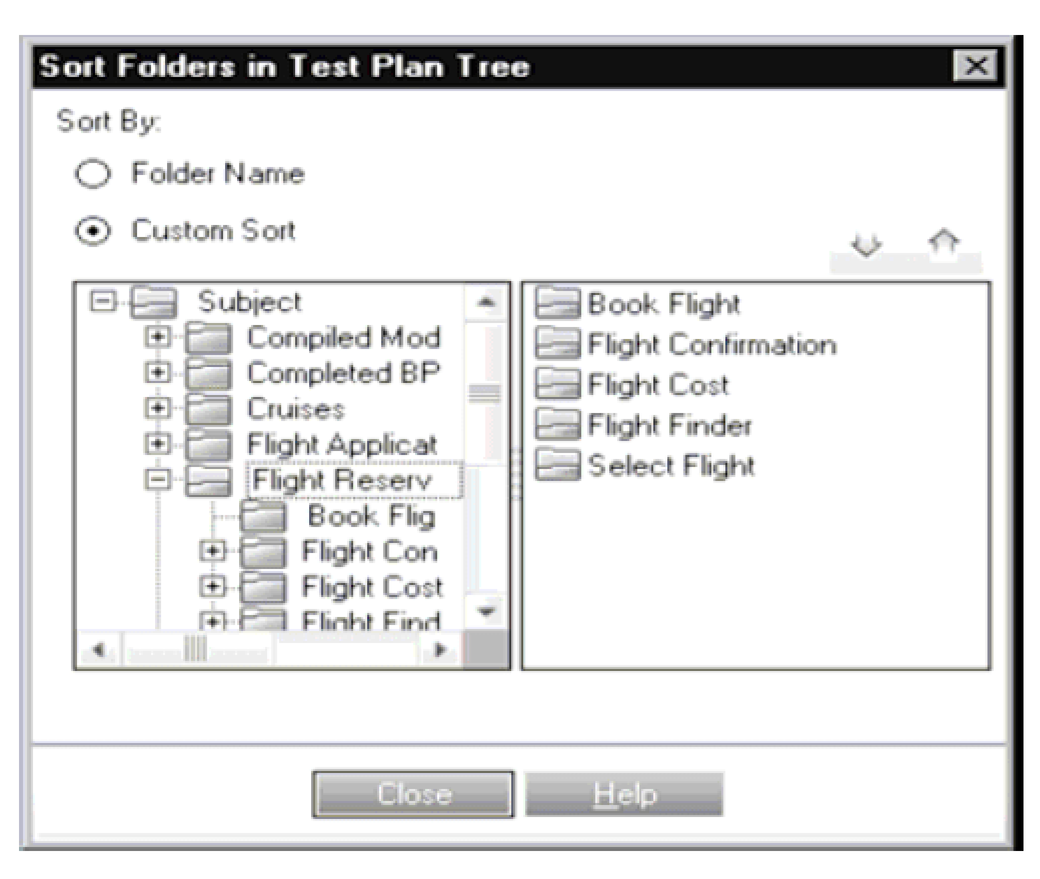


图 11-13　文件夹归类

在测试计划模块中创建测试计划树，可以手动创建文件夹并添加测试到这些文件夹，如图 11-13 所示。测试计划树默认从主题文件夹开始。在主题文件夹中， 你可以创建主文件夹并为每个主文件夹添加子文件夹。

添加文件夹：

1) 从测试计划树中选择 Subject 文件夹来创建主要的主题文件夹(如图 11-14)。

注意：你可以选择存在的主文件夹来创建子文件夹。

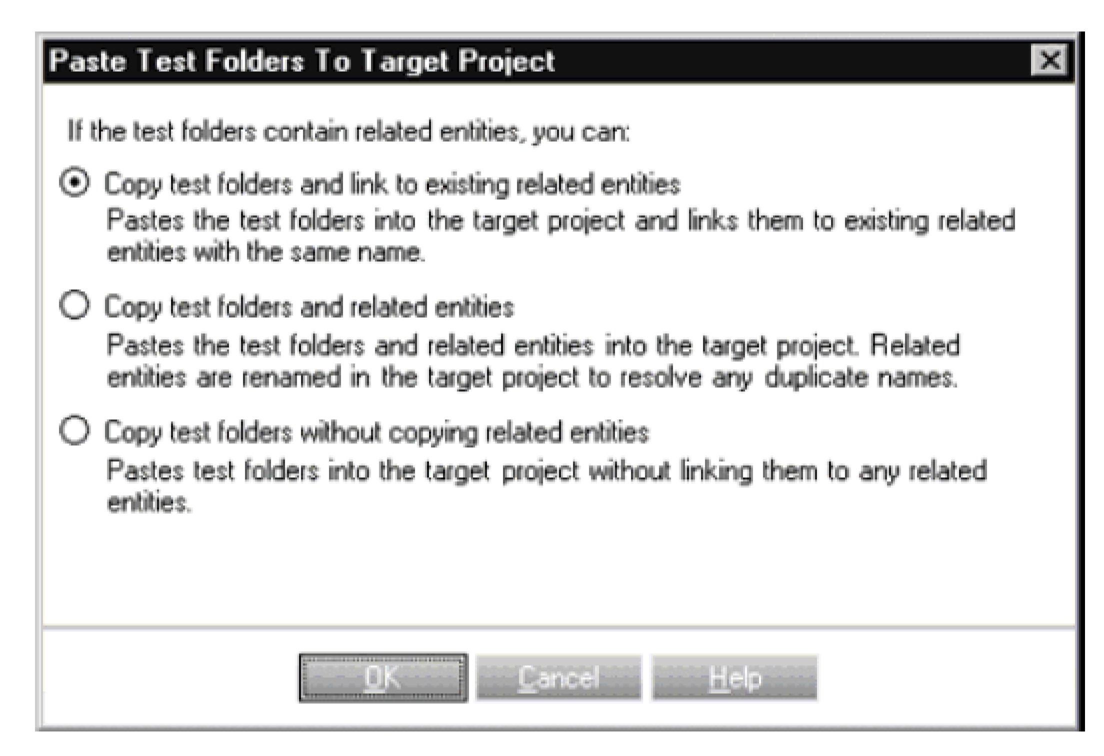


图 11-14 粘贴测试文件夹到目标项目

2) 单击 ALM 工具栏里的，单击 New Folder 按钮，弹出 New Folder 对话框。

3) 在 Test Folder Name 区域中，填写新测试主题的名字。

注意：文件夹的名称不能包含下列任一字符：\、^或*。

4) 单击 OK 按钮，添加文件夹到测试计划树。

注意：开发和编辑测试计划树需要相应的用户权限。

11.5 定义测试的关键点

在测试计划树中定义完主题后，下一步是为每个主题定义要执行的测试。编写的测试需要包含输入、事件、行动和期望的结果，以此来检验程序是否满足指定的需求。在创建测试时，你要确保测试是：

1) 精确的：每个测试都应该有明确的主题，例如检验指定的功能或系统需求。

2) 简短的：一个测试应该只包含为实现目标必定需要的步骤和字段。

3) 一致的：每个重复性的测试都必须一致的执行。

4) 恰当的：一个测试对测试人员和测试环境来说必须是适用的。

5) 可追踪的：测试应该与需求或过程匹配。

6) 定义测试时：(1)添加测试到测试计划树；(2)为测试指定详细信息。

11.6 添加测试

添加测试到测试计划树时，你需要定义关于测试的一些基本信息，例如名称和类型。

1) 从测试计划树中选择想要添加新测试的主题文件夹。

2) 在 ALM 工具栏中单击 New Test，弹出新建测试的对话框。

3) 从 Type 列表中，为测试选择一个类别。

4) 在 Test Name 一栏，为测试编辑一个名称。

5) 注意：测试的名称不能包含下列字符：\、/、：、“、？、<、>、|、*、%或‘。

6) 单击 OK 添加到测试计划树。

11.6.1　测试类型

在计划你的测试时，可参考表 11-2 具体的测试类别。

注意：QUICKTEST_TEST 这个类别只有在 ALM 安装了相应的插件才能使用。

表 11-2　测试类别

按钮图标	测试类型	描　述
	BUSINESS-PROCESS	业务流程测试
	LR-SCENARIO	由 HP 负载测试工具 LoadRunner 执行的场景
	QUICKTEST-TEST	由 HP 企业级功能测试工具 Unified Functional Testing 执行的 GUI 测试。 此类测试只有从 HP Application Lifecycle Management 插件页安装了相应插件后才可用
	VAPI-XP-TEST	由 Visual API-XP(ALM Open Test Architecture API 测试工具)创建的测试
	ALT-SCENARIO	Astra 负载测试场景，在旧的应用系统依然可用
	QAINSPECT-TEST	由 HP 安全测试工具 QAInspect 执行的测试
	MANUAL	手动运行的测试
	SYSTEM-TEST	指示 ALM 提供系统信息、捕获桌面图像或重新启动计算机的测试
	FLOW	由一组顺序固定的业务组件组成用以执行特定任务的测试

11.6.2　公布细节标签

可使用详细标签来定义关于测试的基本信息，例如它的状态、创建时间和设计人员。另外还提供一些标准的文本框。你可以用 Description 框来提供一个测试的简短说明。例如图 11-15 显示的 Description 框包含关于测试的一些叙述。

注意：你可以通过在附件标签添加附件来提供测试的一些额外信息。

如果测试计划树的测试名前面有感叹号，表示这个测试存在告警。红色的感叹号表示这个告警是刚触发的；灰色表示这个告警已经被阅读过。站点管理员可以定义并激活告警规则，创建告警在项目出现变更的时候发送邮件。

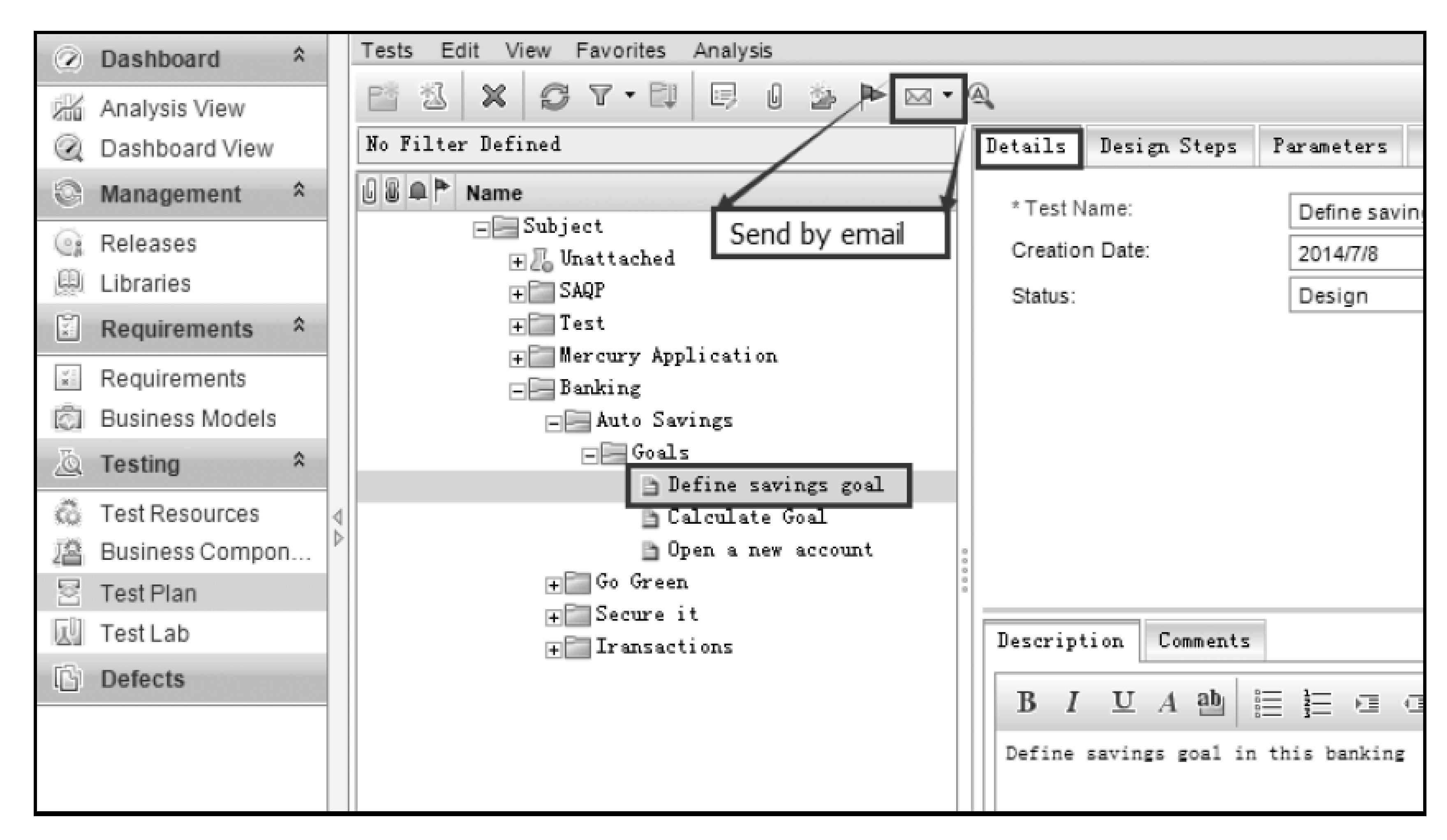


图 11-15　测试细节标签

11.6.3　添加测试步骤

定义完手动测试后，你需要指定执行每步测试的详细步骤。添加步骤需要指定程序执行的行为、输入的值和期望的结果。如图 11-16 所示添加和定义测试步骤。

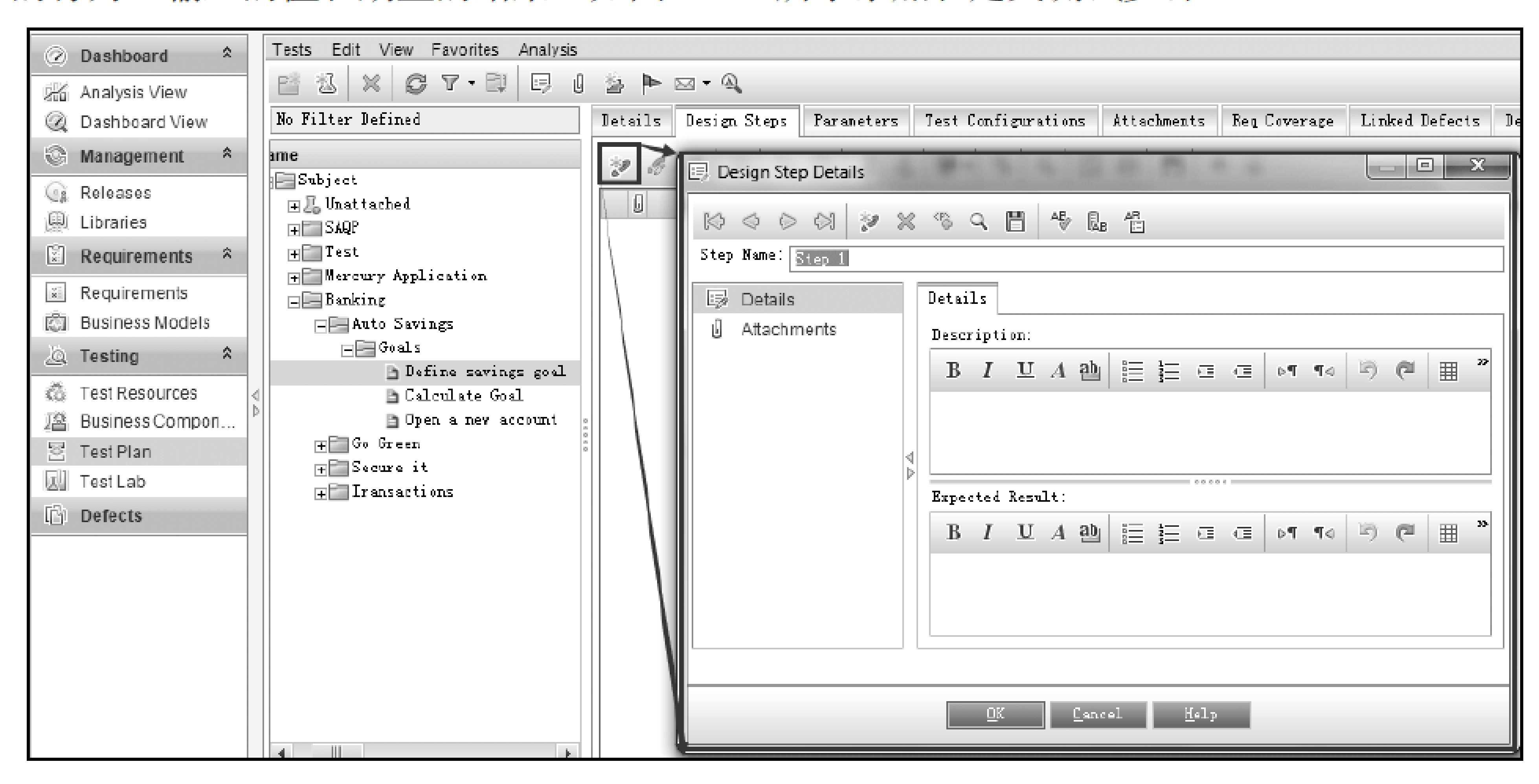


图 11-16　添加测试步骤

1) 从 Test Plan Tree 中，选中一个测试；
2) 在右边的面板中，单击 Design Steps 标签；
3) 单击 Design Steps 工具栏里的 New Step 按钮，弹出 Design Step Details 对话框；
4) 在步骤名对话框中，输入 Step Name；
5) 在 Description 框中，编辑这一步的说明；
6) 在 Expected Result 框中，可以编辑步骤完成之后的期望结果。另外测试人员也可以使

用指定的详细说明来检验结果；

7) 单击 OK，测试步骤出现在 Design Steps 页面。足迹图标显示测试计划树中下一个测试的名字，这个图标表明了为测试定义的步骤。

11.6.4　设计测试步骤的注意事项

在设计测试步骤时，你需要定义程序的操作来确保程序的各方面都被测试到。为了确保你清楚、准确地捕获到了完成一个操作所需要的所有行为，必须满足以下条件：

1) 每一步都用主动语态书写。使用主动语态的好处是使执行测试的人清楚每一步要如何来执行。

2) 每步用一个动作并且清楚地描述行为是否应该被测试人员或程序执行。

3) 确保没有遗漏步骤。

4) 整个测试都使用一致的术语。

5) 检验测试中存在的字段并保持与被测系统中相同的标记。

6) 为测试指定通过和失败的条件。

11.7　调用测试

建立测试的步骤来包含对其他测试的调用。这样可以形成模块化并能在多个测试之间重复利用一个标准的有序步骤。例如，图 11-17 显示了定义储蓄目标测试包含了一个对预算目标的测试的调用。

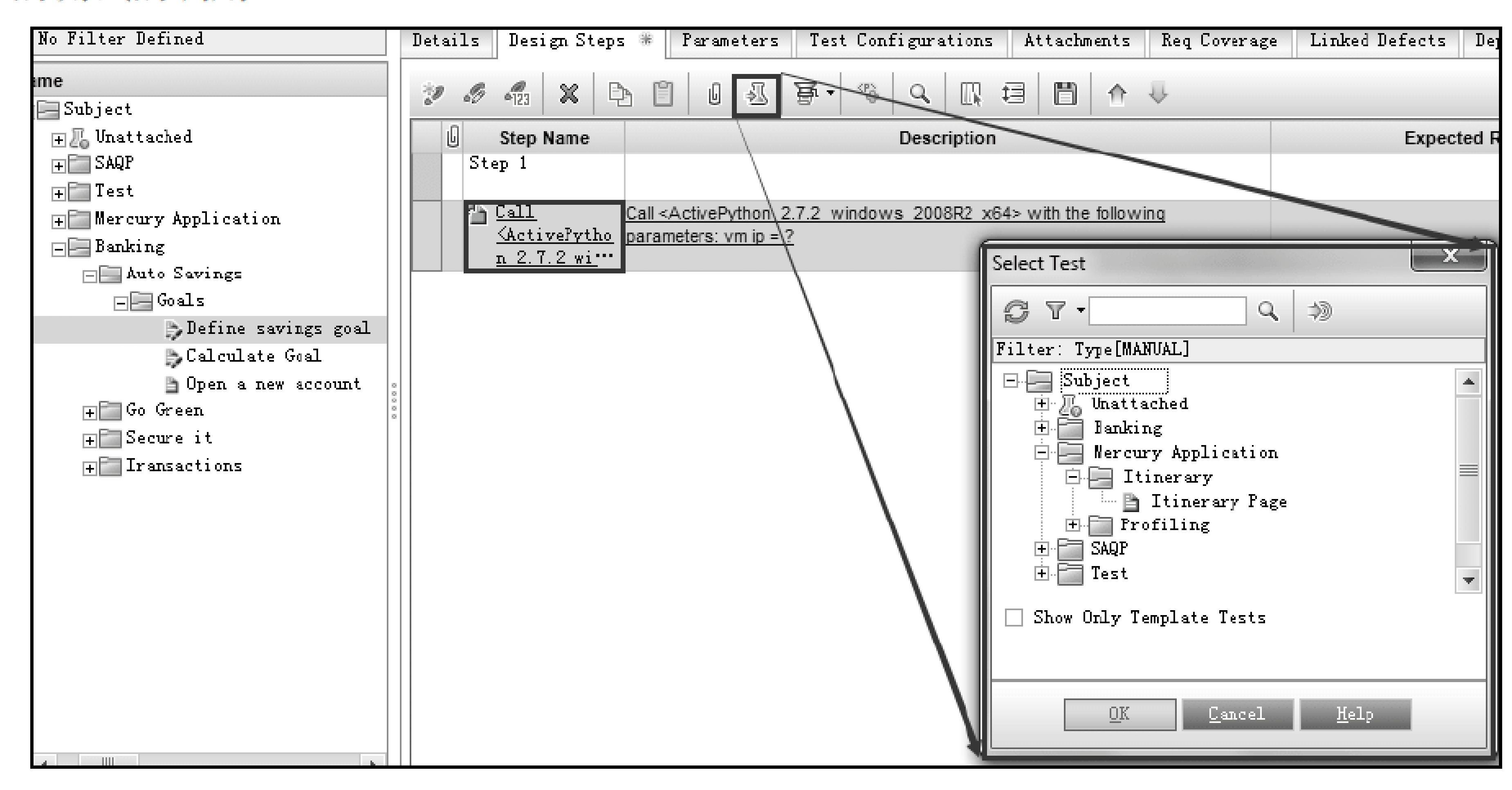


图 11-17　调用测试

在测试中一个步骤调用另一个测试：

1) 单击调用测试的 Design Steps 标签。

2) 在 Design Steps 标签页面，单击 Call To Test 按钮。Select Test 对话框出现。

3) 选择要调用的测试并单击 OK。这样就在当前测试中添加了一个步骤并标记为“Call <TEST_NAME>”。如果被调用的测试有未赋值的参数，测试参数对话框出现，可供配置参数。

11.8　测试参数

参数是执行测试期间变量分配的值，在调用测试时通过动态地改变参数的值让测试更加灵活。运行时，可对参数指定数据值来控制测试的执行。

如图 11-18 显示的是 Calculate Goal 测试的第一步使用了一个参数 Amount Type。当另一个测试需要运行计算目标函数时，它可以调用 Calculate Goal 测试并对这些参数赋予相应的值。Calculate Goal 测试定义的参数有：<<<Amount Type>>>、<<<Amount to Save>>>、<<<Goal Month>>>、<<<Goal Year>>>、<<<Acct Type>>>，你可以在测试步骤的 Description 和 Expected Result 中使用参数。

11.8.1　定义参数

在参数标签下创建和显示参数。参数对话框则用来将测试参数插入到测试步骤。如图 11-19 所示。

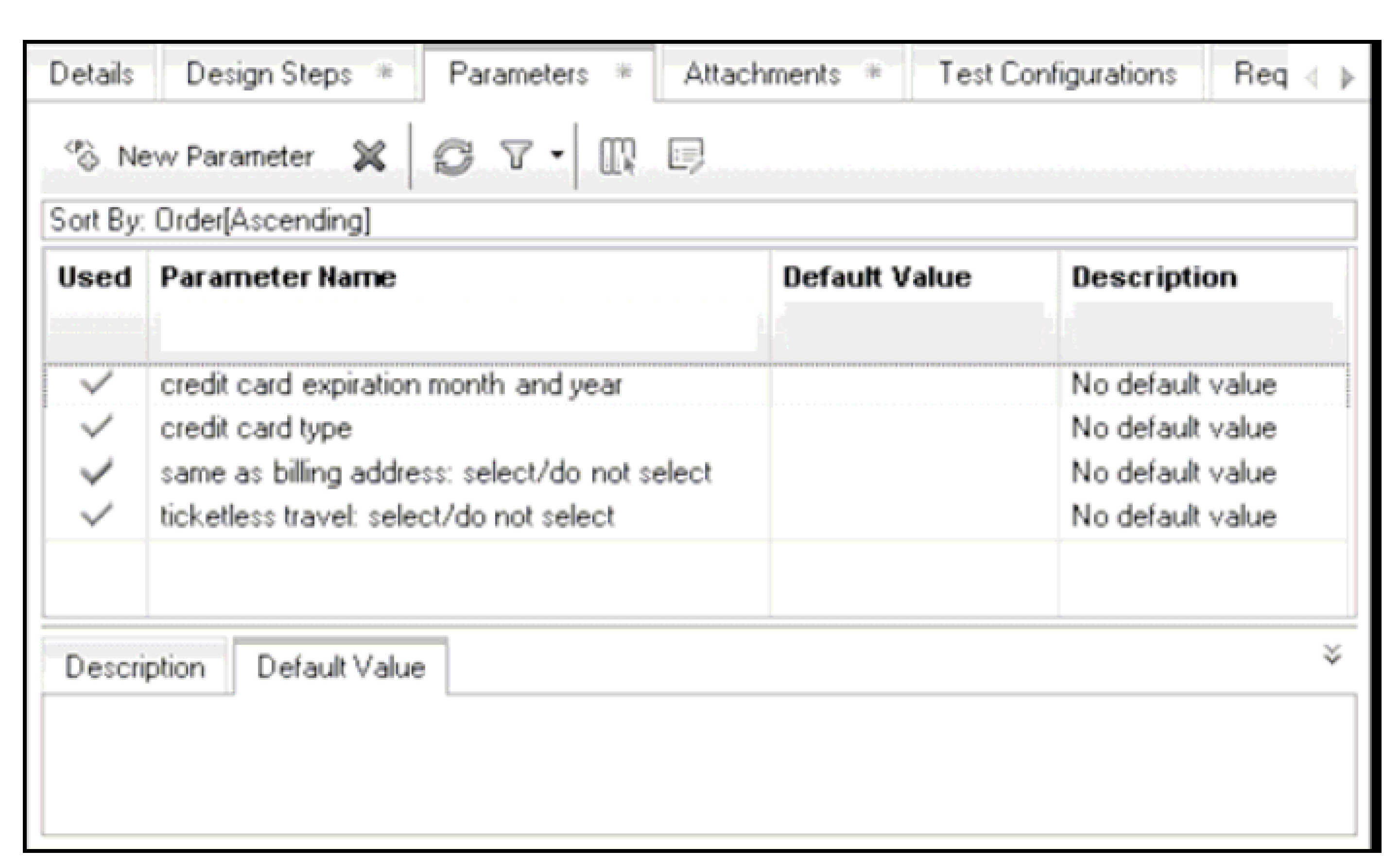


图 11-18　测试参数

定义一个参数：

1) 从测试计划树中选中一个测试；

2) 单击 Design Step 标签；

3) 在设计步骤页面的工具栏中，单击 New Step 按钮，出现 Design Step Details 对话框；

4) 在 Description 或 Expected Result 中找到你想要定义参数的步骤；

5) 单击 Insert Parameter 按钮，出现 Parameter 对话框。单击 New Parameter，出现 New Test Parameter 对话框；

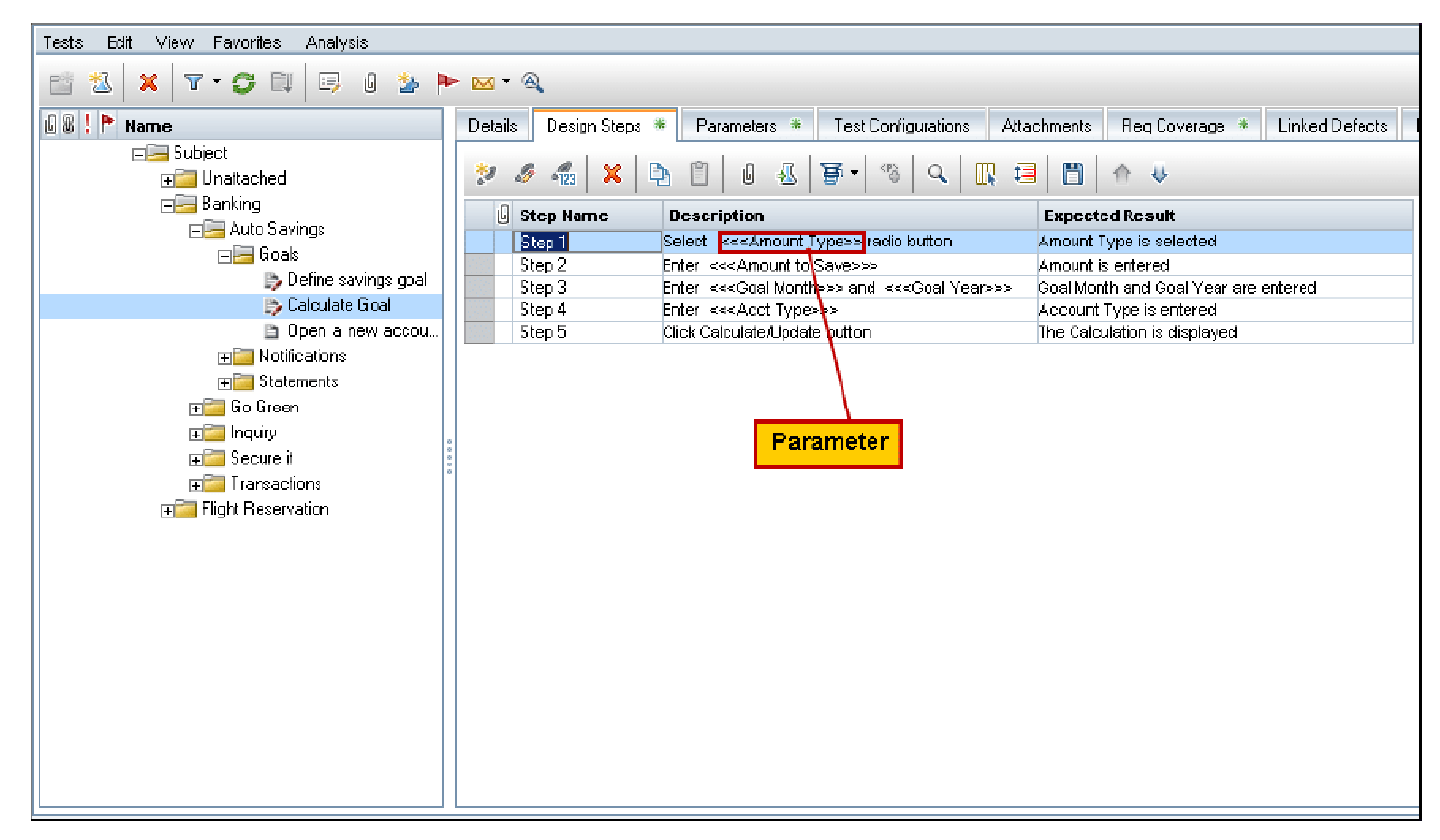


图 11-19　参数标签对话框

6) 在 Parameter Name 一栏，输入一个参数名然后单击 OK，单击 Parameter 对话框的 OK，新参数将以<<<parameter_name>>>添加到步骤中；

7) 注意：参数名不能包含下列任一字符：~、?、 ‘、<或>；

8) 单击 OK 关闭 Design Step Details 对话框。

11.8.2　调用带有参数的测试

当调用包含参数的测试时，你可以设置想传给参数的值。图 11-20 中 Define savings goal 测试就是调用 Calculate Goal 测试并将指定的值传给 Calculate Goal。

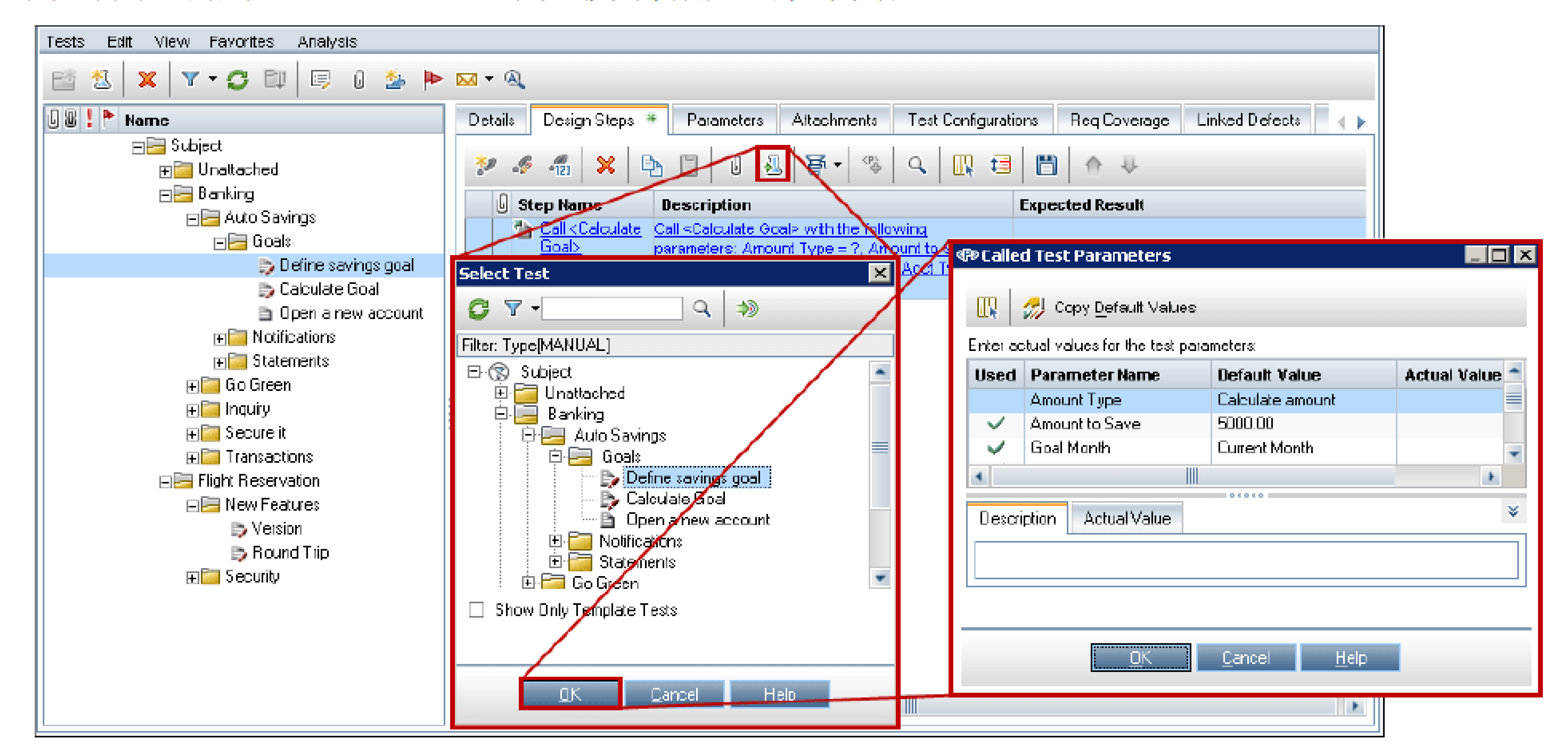


图 11-20　调用带有参数的测试

如何调用一个测试并为参数传值：

1) 从测试计划树中选中需要调用的其他测试的测试；

2) 单击测试的 Design Step 标签，单击 Call To Test 按钮，将打开 Select Test 对话框；

3) 选中你想要调用的测试；

注意：如果选中 Show Only Template，在 Select Test 对话框中将只显示模板测试，模板测试是包含步骤和参数的测试，这些步骤和参数通常可以在不同的测试中反复使用。但是，从其他测试调用它之前没必要将测试转换为模板。

4) 单击 OK，将出现 Called Test Parameter 对话框；

5) 在调用的测试中编辑想要传给参数的值；

6) 单击 OK，新添加的步骤将包含调用选出来的测试和需要传给测试参数的值。

11.8.3　编辑被调参数值

在定义完测试调用之后，你依然可以对赋予参数的值进行修改。例如在图 11-21 中，你可以对 Calculate Goal 测试步骤中参数赋予的值进行修改。

1) 右键单击调用的步骤，单击 Called Test Parameter，将出现 Called Test Parameter 对话框。

2) 单击参数的 Actual Value 这一列，编辑一个新的值。

3) 单击 OK，测试调用将更新为新的值。

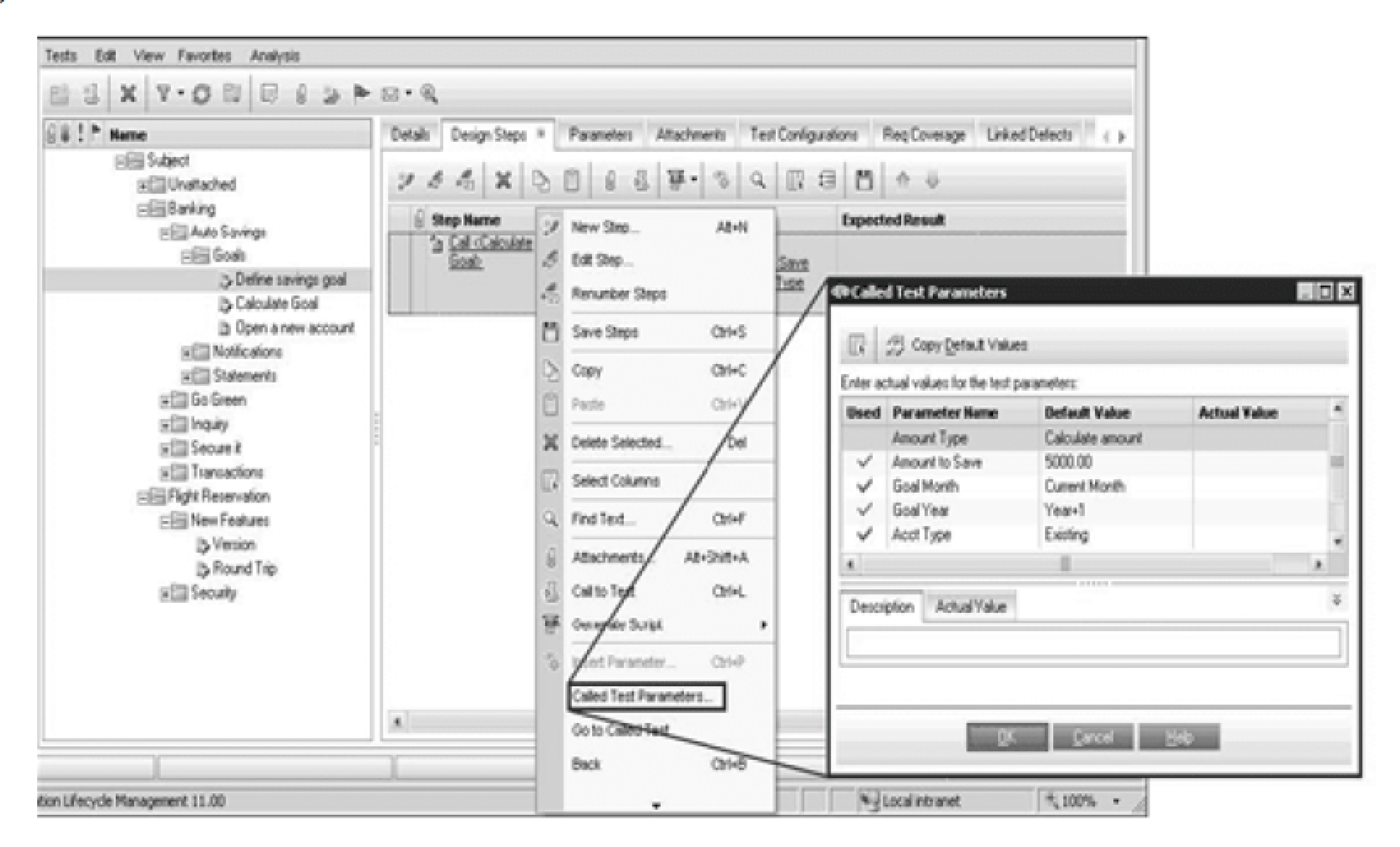


图 11-21　编辑被调参数

11.9　创建测试模板

每个存在的手动测试都可以被复制，以此来作为新建一个手动测试的基础。将一个手动测试标记为模板测试会有一个特殊的名称。 但是模板测试在功能上和其他手动测试没有区别。当对话框中的 Show Only Template Test 复选框选中时，ALM 就会使用标记的名称。如图 11-22 所示。

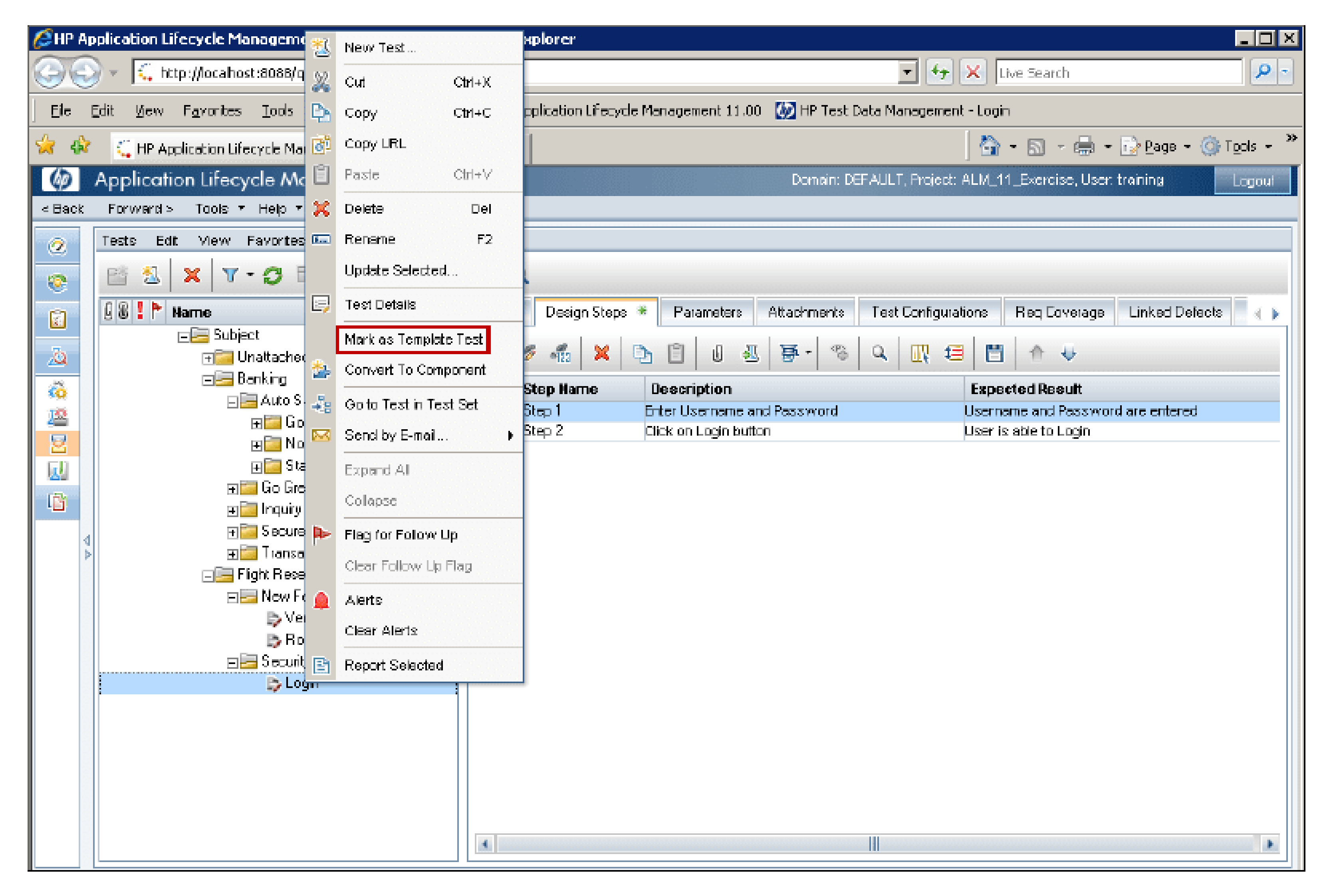


图 11-22　创建测试模板

将测试配置为模板测试：

1) 从模板树中，右击一个手动测试。

2) 选择 Mark as Template Test。

3) 当对话框中的 Show Only Template 被选中时，没有模板测试的手动测试将从可以测试列表中被过滤掉。

11.10　创建测试脚本

一个测试脚本包含测试执行需要完成的操作。可以从手动定义的测试步骤中生成自动测试脚本来完成自动测试。根据这些测试步骤，ALM 可根据你选择的自动测试工具来创建自动测试脚本模板。例如，图 11-23 显示的就是根据 Define Saving Goal 测试创建的 QTP 脚本。

将设计步骤转换测试脚本：

1) 单击 Design Step 选项卡；

2) 在 Design Step 的工具栏中，单击 Generate Script 按钮，将出现一个下拉菜单；

3) 选择你用来记录业务流程和完成测试的自动测试工具；

4) 生成脚本后，Test Script 选项卡会出现一个星号，单击 Test Script 选项卡来显示生成的测试脚本。

注意：测试脚本生成后，测试计划树中的手动测试图标会替代成和你选中的自动测试的图标一样。单击测试脚本页面的 Launch 按钮，可以直接从创建的测试工具中打开和修改测试脚本。

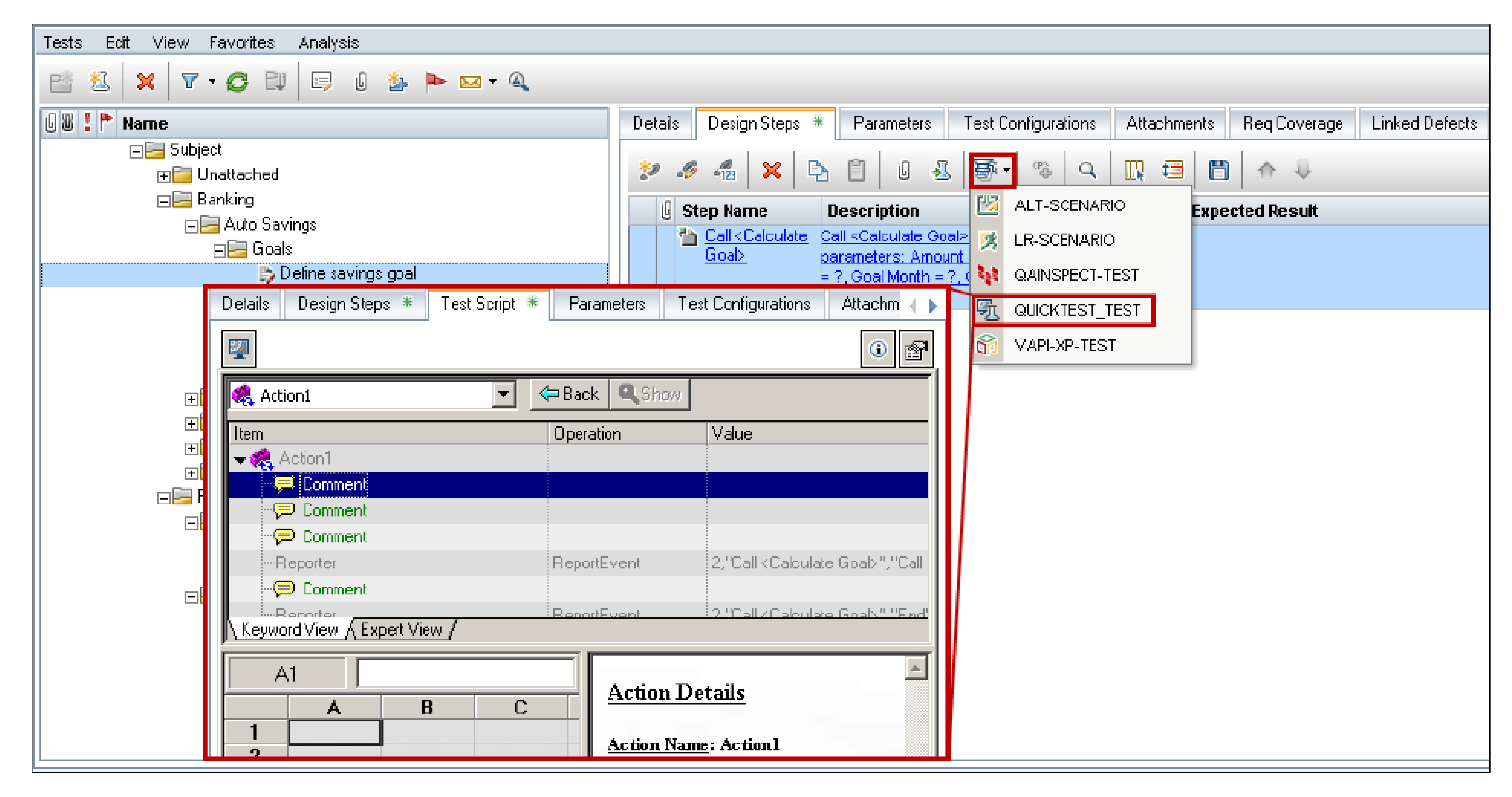


图 11-23　创建测试脚本

11.11　测试配置

现在你可以设计测试并通过不同的使用案例、不同的数据集来运行测试。一个使用案例就称为一个测试配置，而测试配置的值则由 ALM 项目或外置的数据源来提供。

创建测试时，HP ALM 会默认创建一个测试配置。这个测试配置使用和测试一样的名称创建。如图 11-24 所示。

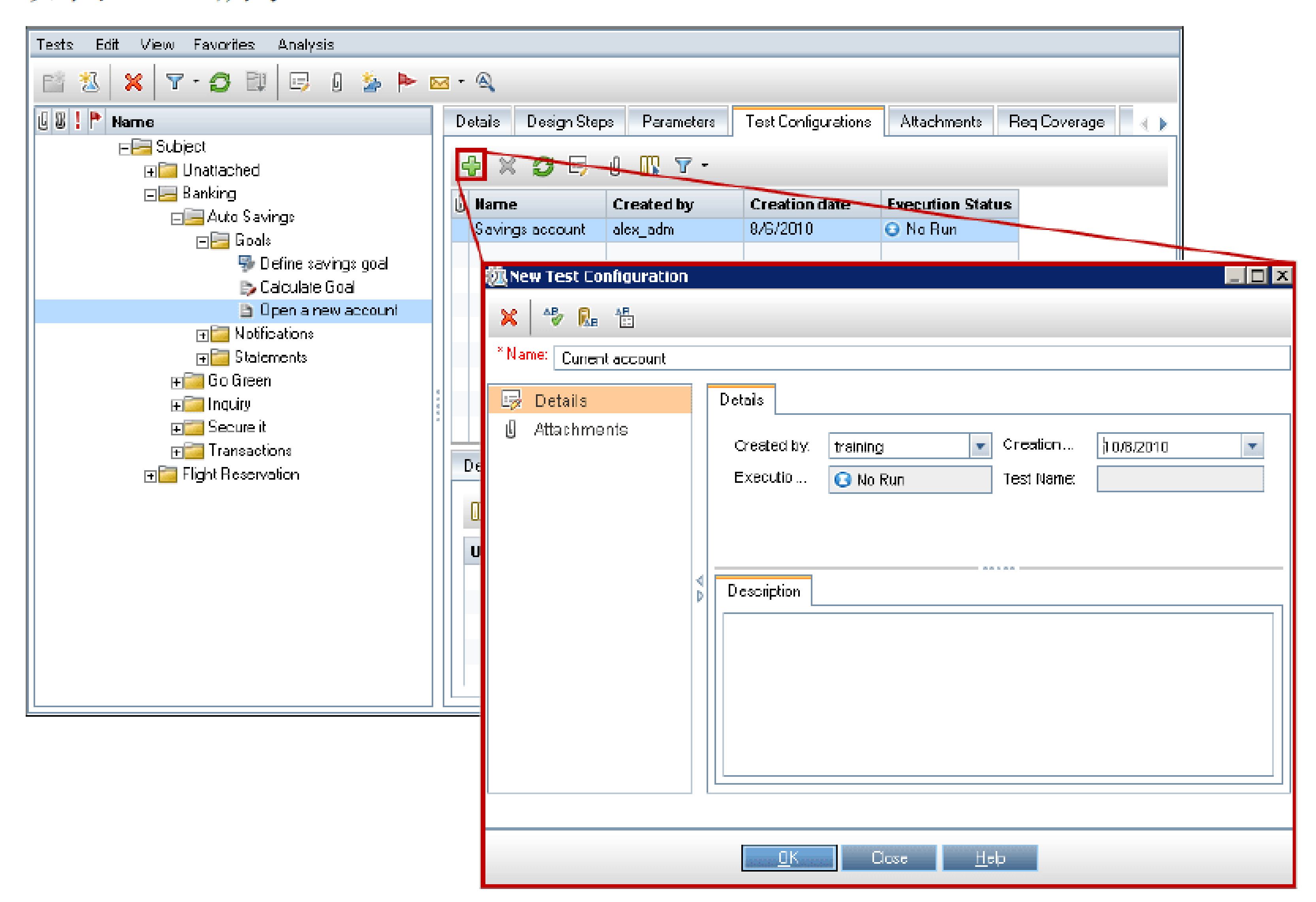


图 11-24　测试配置

测试配置概述：

1) 测试配置是一系列描述测试的具体使用案例的定义；

2) 可为每个测试配置关联不同的测试集；

3) 使用测试配置你可在不同环境下运行同一个测试；

4) 在创建测试时，HP ALM 会默认创建一个测试配置。这个测试配置使用和测试一样的名字创建；

5) 在测试计划模块的 Test Configuration 选项卡，你可以根据需要创建许多额外的测试配置；

6) 可将测试配置与测试计划模块下参数选项卡中数据相关联，还可将每个测试配置与不同的数据关联。

11.11.1　定义测试配置

图 11-25 显示了如何定义测试配置。

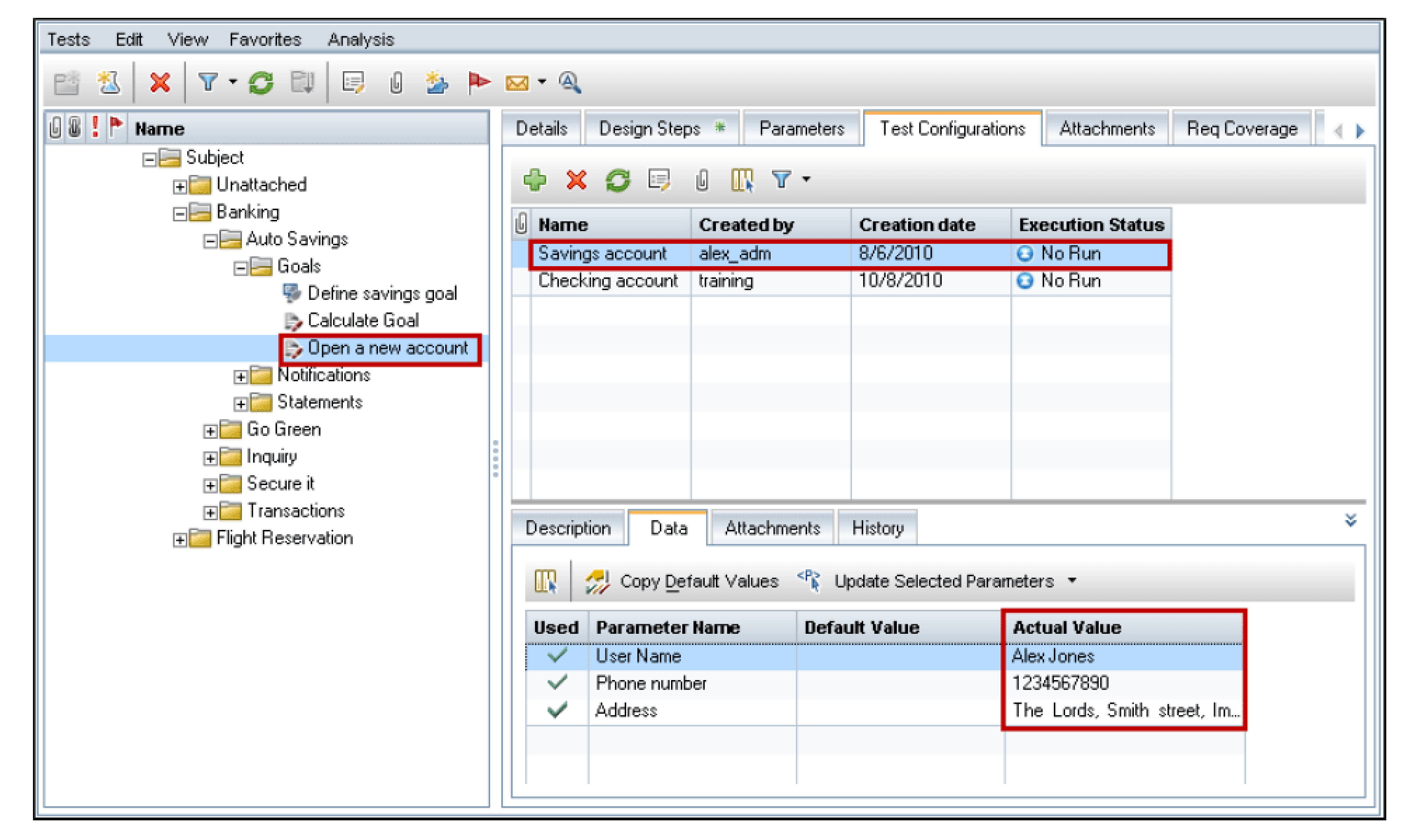


图 11-25　定义测试配置

1) 在测试计划树中，选中一个测试，然后单击 Test Configuration 选项卡；

2) 选中一个存在的配置，单击 Test Configuration Details 按钮；

3) 在 Details 选项卡中，改变配置的名称，然后单击 OK 按钮；

4) 在 Data 选项卡中，通过单击 Parameter Name 的 Actual Value 这一栏的下拉框为每个参数都输入一个有效值；

5) 创建一个新的测试配置，单击 New Test Configuration 按钮，将出现 New Test Configuration 对话框；

6) 为新建的测试配置取个名称，单击 OK；

7) 在 Data 选项卡下，通过单击 Parameter 的 Actual Name 这一栏的下拉框为每个参数输入一个有效的值。

11.11.2　测试配置窗口

图 11-26 的窗口列出的是选中测试的配置。

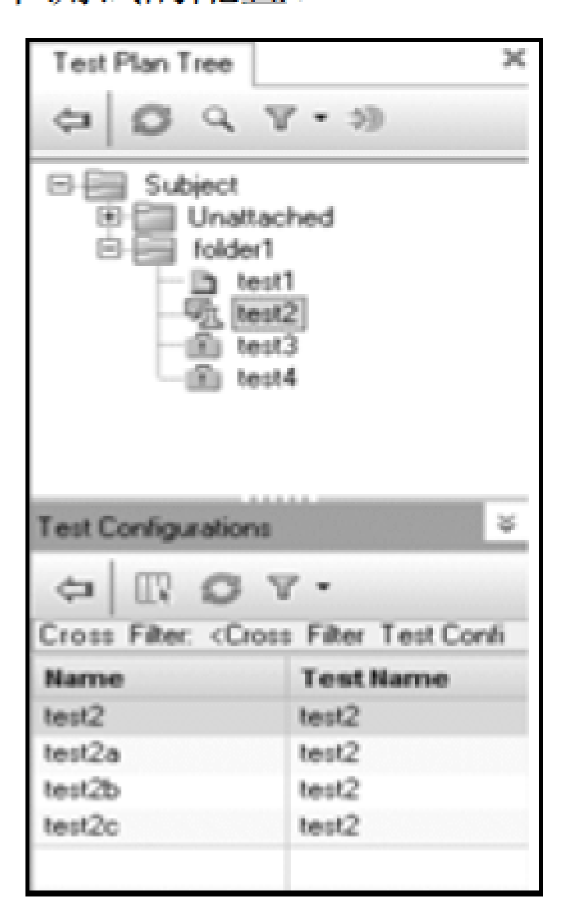


图 11-26　测试的配置

11.11.3　将测试配置添加到需求范围

将测试配置添加到需求范围(如图 11-27 所示)。

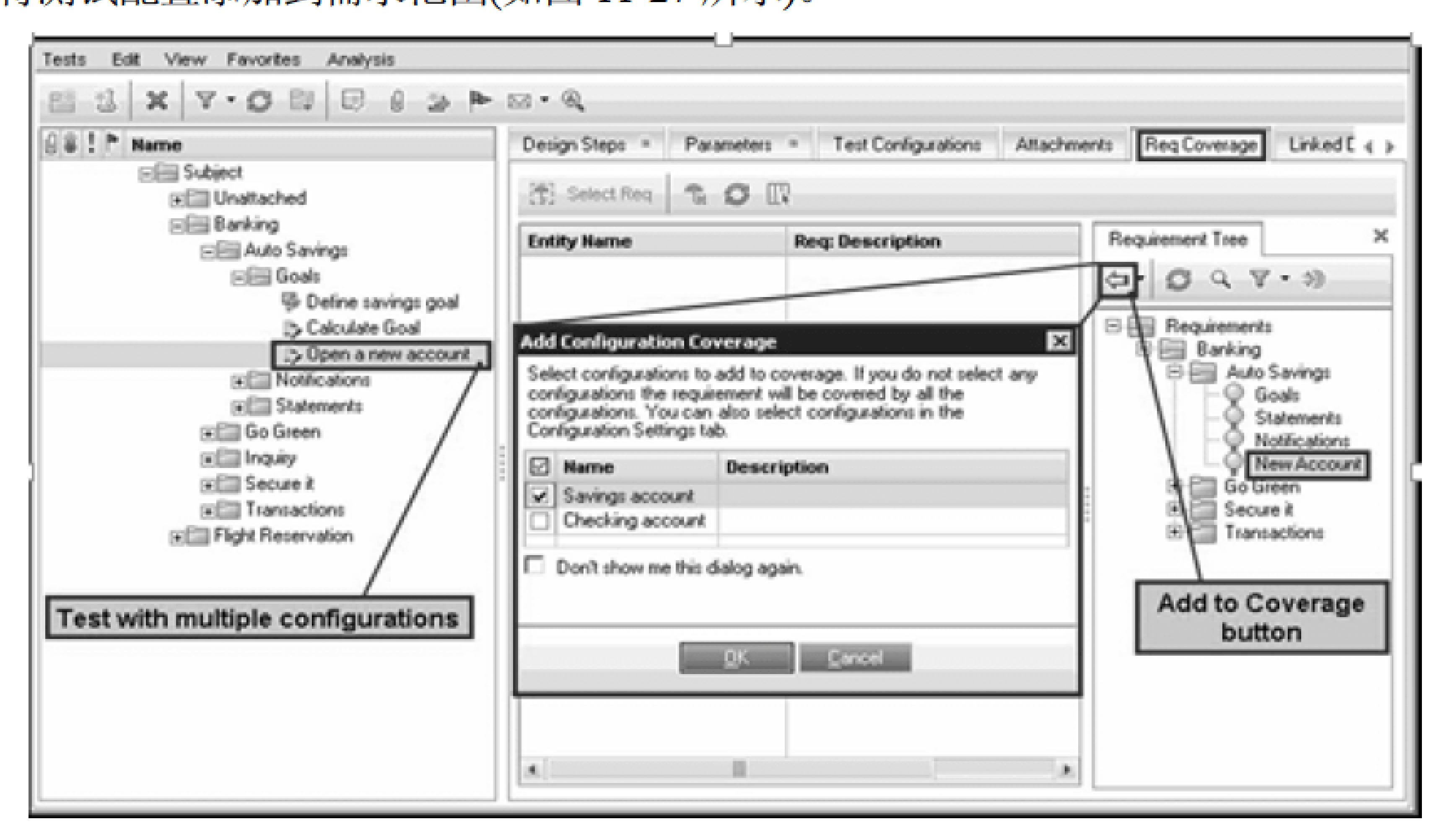


图 11-27　添加配置范围

1) 在测试计划树中，选中一个测试和多个测试配置；

2) 单击 Req Coverage 选项卡；

3) 单击 Select Req 按钮，右边将出现 Requirement Tree；

4) 选中合适的需求，单击 Add To Coverage 按钮，将出现 Add Configuration Coverage 对话框；

5) 选择具体的配置，选中后单击 OK 按钮，如果你没有选中任何配置，需求将涵盖整个配置；

6) 单击 OK 按钮。

11.12　实时分析图表

你可从测试计划模块中生成一个实时分析图表，实时分析图表对测试计划树文件夹下的所有测试提供了一个虚拟综述。当你在测试文件夹下更新一个测试时，数据的变化也会反映到图表上。另外，当选中测试计划模块下的另一个测试文件夹时，图表的布局和设置会被隐藏。这种特征可以让你在不同的文件夹下看到一样的图像分析，而不需要重绘图表。

生成实时分析图表(如图 11-28 所示)：

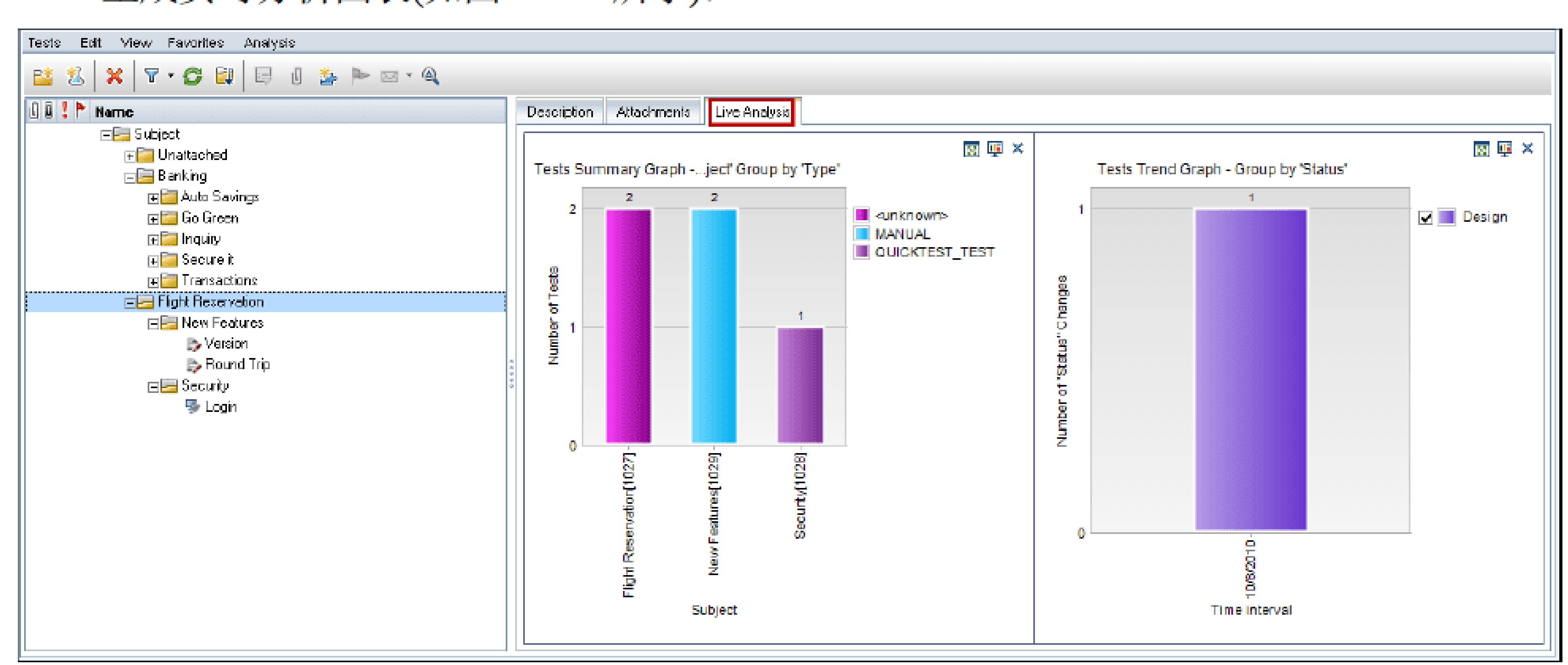


图 11-28　实时分析图表

1) 从测试计划树中选中一个测试主题文件夹；

2) 单击 Live Analysis 选项卡；

注意：Live Analysis 选项卡分为两个窗口，在想要显示图标的窗口单击 Add Graph 链接。将出现图像向导：两步中的第一步。

如果已经显示了两个图表，但是还想再创建一个新的图表，则需要删除一个存在的图表。删除图表的步骤如下：(1)单击你想要删除的图表顶部的 Delete Graph 按钮；(2)单击 YES 确认图表已经从选中的窗口中删除，并且出现了 Add Graph 的链接。

3) 单击 Add Graph 链接，在 Graph Type 处选择你想要生成的类别。你可以生成下面这些类别：(1)汇总：显示选中的测试主题文件夹下存在的测试数量；(2)进展：从一段时间中指定一个时间，并显示当前时间选中的测试主题文件夹下存在的测试数量；(3)趋势：从测试计划模块中指定一个字段，显示该字段在特定时间间隔内的变更历史；

4) 单击 Next；

5) Group By Field下拉框中，选择一个想用来在图表中对测试进行分组的字段，单击Next；

6) 在 X-axis field 下拉框中，选择一个作为 X 轴的字段；

7) 单击 Finish，图标将出现在选中的面板中。

11.13　修改图表外观

生成实时分析图表后，可根据自己的需求来修改实时分析图表的外观。例如，你可以改变在 X 轴想用的字段。如图 11-29 所示。

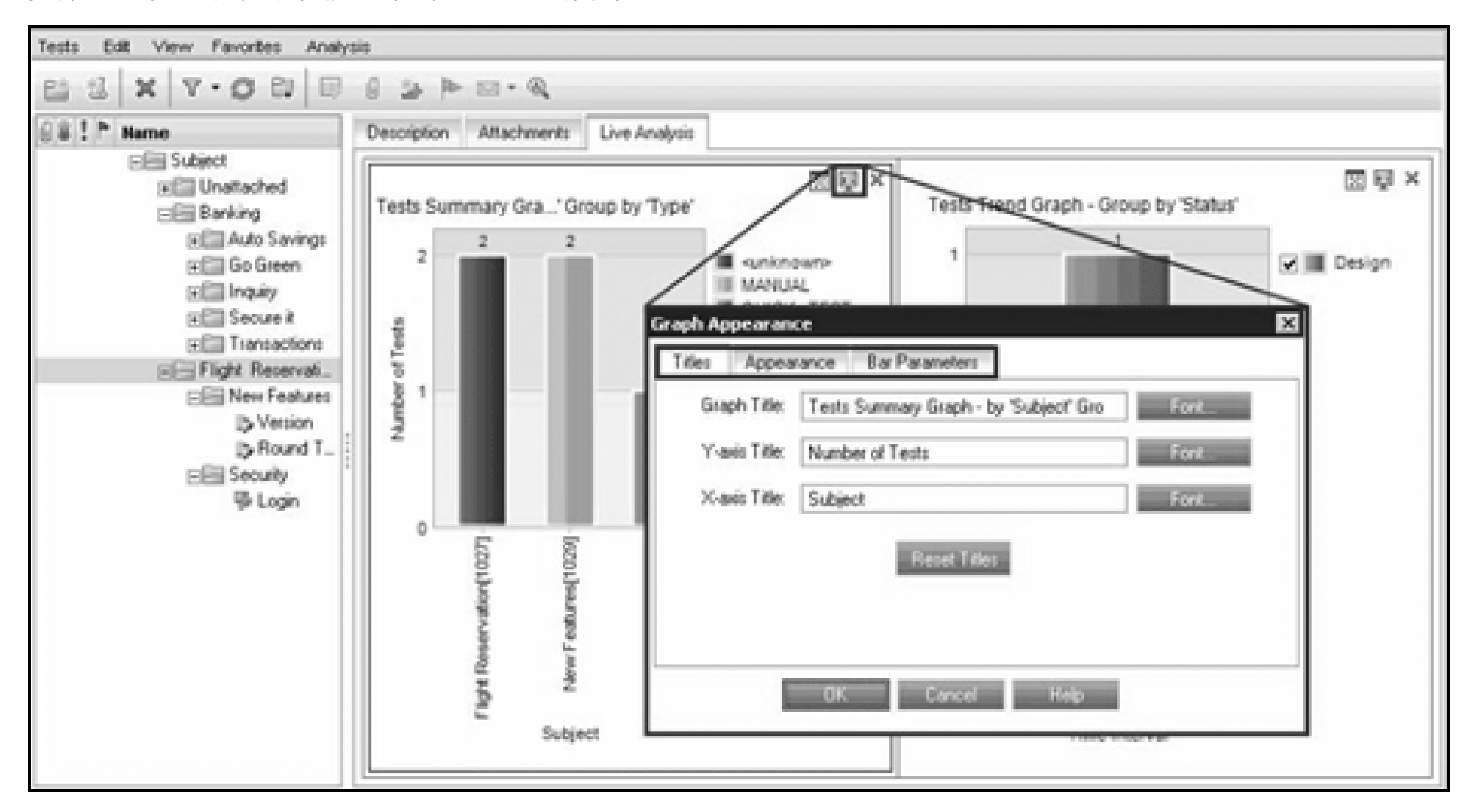


图 11-29　修改实时分析图表

修改实时分析图表的外观：

1) 在图表所在的窗口，单击 Set Graph Appearance。将出现 Graph Appearance 对话框。在 Graph Appearance 对话框中，默认打开 Title 选项卡，其中会显示 Graph Title、Y 轴标题、X 轴标题。可修改这些值。单击 Reset Titles 按钮将返回到初始值；

2) 单击 Appearance 选项卡，在 General 处选择处选择 Default Layout、Legend Position、3D Graph 和 Vertical X-axis Labels；在 Colors 处修改标签的颜色；

3) 单击 Bar Parameter 选项卡来修改图表中标记的位置和风格；

4) 单击 OK 关闭 Graph Appearance 对话框。

习题与思考题

1. 测试过程中，测试计划模块最重要的是什么？
2. 实时分析图表的作用是什么？
3. 什么是模板测试？
4. 什么是需求的直接范围状态？

练习：搭建一个测试

在需求间添加追踪链接并分析风险与需求之间的关系后，你需要创建测试来检验是否满足需求。如果你决定将需求转换为创建一系列测试计划，要注意尽管测试计划是从需求转换来的，并且自动与需求关联，但并没有测试步骤。

本练习中，将通过添加测试步骤、参数化和标记为模板来完成转换测试。然后能够从其他测试调用可再用模板。

在这个练习中，你需要完成下面一些任务：

第 1 步：将需求转换为测试计划

第 2 步：创建登录测试

第 3 步：定义测试配置

第 4 步：创建打开订单测试

第 5 步：关联需求与测试

(该练习的指导步骤请参见本章正文)

第12章　执行测试

12.1　在 ALM 中运行测试

应用程序生命周期管理进程的第 4 步是将测试纳入测试集，运行测试(如图 12-1 所示)。测试运行后，就完成了列举程序中的不一致、问题以及缺陷等内容的文档。之后可将这些问题进一步汇总到缺陷跟踪系统，以便进行深入调查、改正和重新测试。

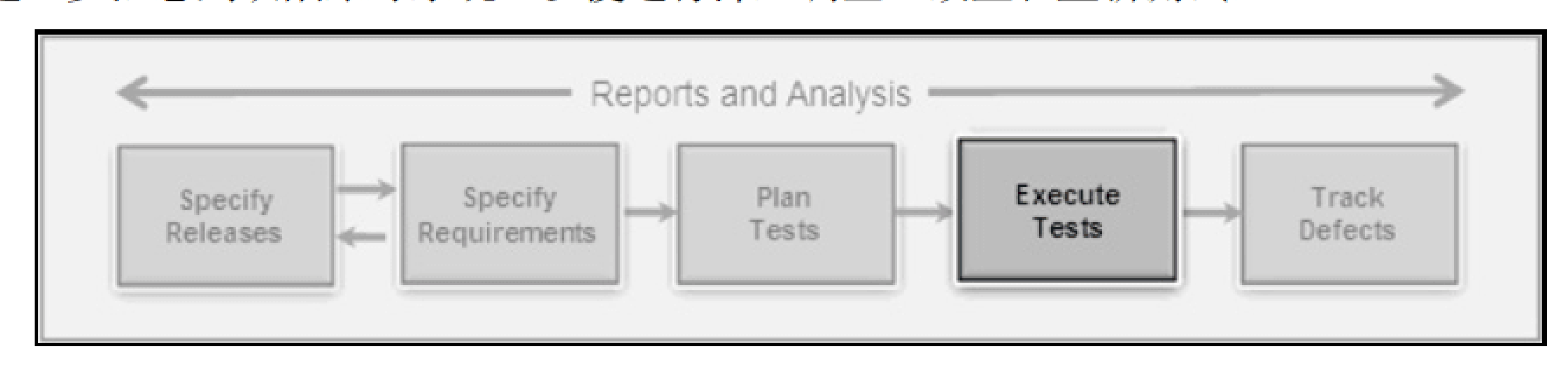


图 12-1　执行测试

通过创建的测试集合和选择测试集合中应包含的测试开始执行测试。在惠普应用程序生命周期管理(ALM)工程中，一个测试集合包含一个测试的子集合，其目的是为了达到指定的测试目标。一旦应用程序发生变更，需要在工程中进行手工和自动化测试，以便定位缺陷和进行质量评估。你可以手动或自动地运行 ALM 的测试。

1. 手动运行测试

1) 惠普 Sprinter。在手动测试过程中，Sprinter 提供强大的辅助功能。ALM 版本：在质量中心初始版本和性能中心版本中不提供 Sprinter 的功能。

2) 手工运行器。如果你没有使用 Sprinter，可以使用手动运行器手动运行测试。可以在 ALM 中手动地运行手工测试和自动测试。手动运行测试时，在应用程序测试中，需要严格按照测试步骤执行应用程序操作，依据实际应用结果和期望输出是否匹配判定每个步骤是否通过。

2. 自动运行测试

1) 应用自动运行器自动运行测试。

2) 可以自动地运行手工测试和自动测试。

3) 自动运行自动测试时，ALM 自动打开所选的测试工具，在本机或远程主机运行测试，向 ALM 输出结果。

4) 自动运行手工测试时，ALM 通过邮件通知指定的测试员，到指定主机上进行测试。

3. 如何在 ALM 上进行测试

1) 创建测试集合

在测试操作模块中创建和定义测试集合。在测试集合创建完成后，将测试集合文件夹分配到发布模块发布树中定义的周期内。

2) 安排测试运行进度

设置环境，安排执行测试的日期和时间进度。也可以排列测试执行顺序。

3) 手动运行测试

可以手动地运行手工和自动测试，执行测试计划中定义的测试步骤。

4) 自动运行测试

可以选择测试集合中的手工或自动测试，自动地运行。

5) 执行性能测试

性能中心：通过运行性能测试为应用程序生成负载并测试其性能。

6) 观察分析测试结果

运行测试后，检查结果，判断实际结果与预期测试结果是否一致。

4. 通过创建图表或报表分析运行数据。

1) 观察测试集合文件夹的动态图表。在测试集树中，选择一个测试文件夹，单击生成动态分析表单转换按钮。

2) 观察图表中的测试集数据。在测试操作模块菜单中，选择 Analysis | Graphs。

生成测试集数据报表。在测试操作模块菜单中，选择 Analysis | Reports。

5. 连接缺陷

如果发现一个缺陷，可创建一个新缺陷并将其连接到测试集，测试实例，测试运行，或者测试步骤，或者连接一个已经存在的缺陷。

12.2　使用 Test Lab 模块-测试集合标签

要在 Test Lab 模块中实现所有的测试执行任务。在 Test Lab 模块，将测试纳入测试集。测试集是设计来实现特定测试目标的测试组。完成测试集创建后，将测试集分配到被定义在 Management 模块里的发布中。测试集的目标一定要与发布中分配的测试目标同步。

定义测试集时，ALM 向测试集中添加已选的测试实例。每个测试实例都包含了一个已

定义的测试配置。测试配置使在不同环境下运行同一测试成为可能。

将测试集分配给发布后，需要预定测试集中的测试执行。可为测试执行指定条件和顺序。基于执行条件，在测试集中执行手动测试。ALM 在指定的日期、时间和关系下，自动执行自动化测试。测试执行完成后，分析测试结果，判定缺陷是否应记录为失败步骤。

测试操作模块侧边栏， 在 Testing 下，选择 Test Lab。Test Lab 模块包括两个标签：Test Sets 和 Test Runs。

图 12-2 所示测试集标签可以用来创建测试集，并进行修改。这个标签包含以下元素：测试集树、Details 标签、Execution Grid 标签、Execution Flow 标签、Automation 标签、Attachments 标签、Linked Defects 标签、History 标签。

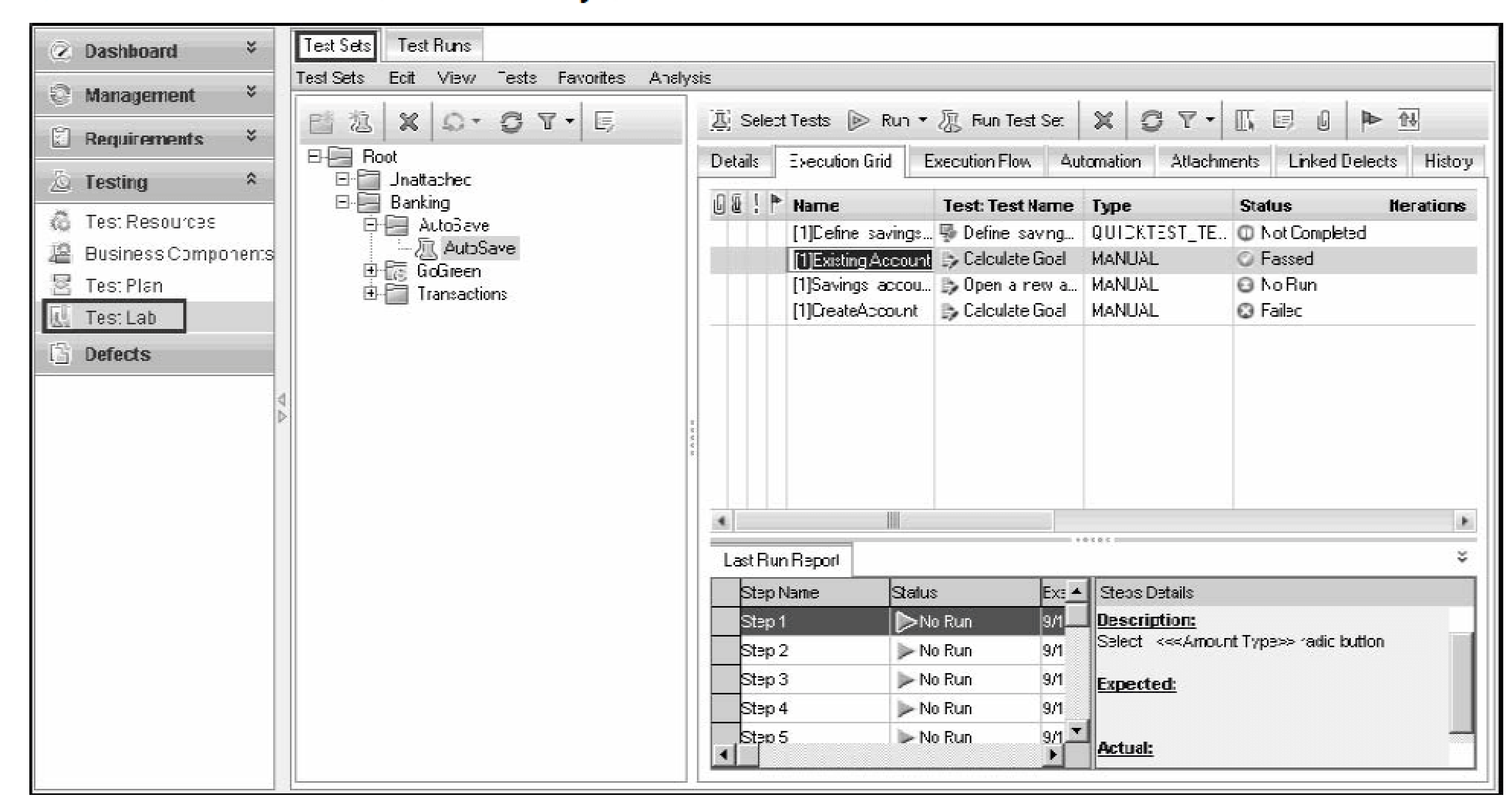


图 12-2　使用 Test Lab 模块-测试集合标签

12.3　Test Runs 标签

Test Runs 标签使你可以在一个网格表中查看所有的项目测试运行。默认情况下，网格表会进行过滤只显示当前公历月中运行的测试，以与事件发生相反的顺序排列(多数将最近发生的事排在最前面)。要去掉这个过滤，去掉在 Exec Date 域中的 This Month 值即可。

如图 12-3 所示，访问 Test Runs 标签。

1) 侧边栏，测试标签中，选择 Test Lab。

2) 单击 Test Runs 标签。

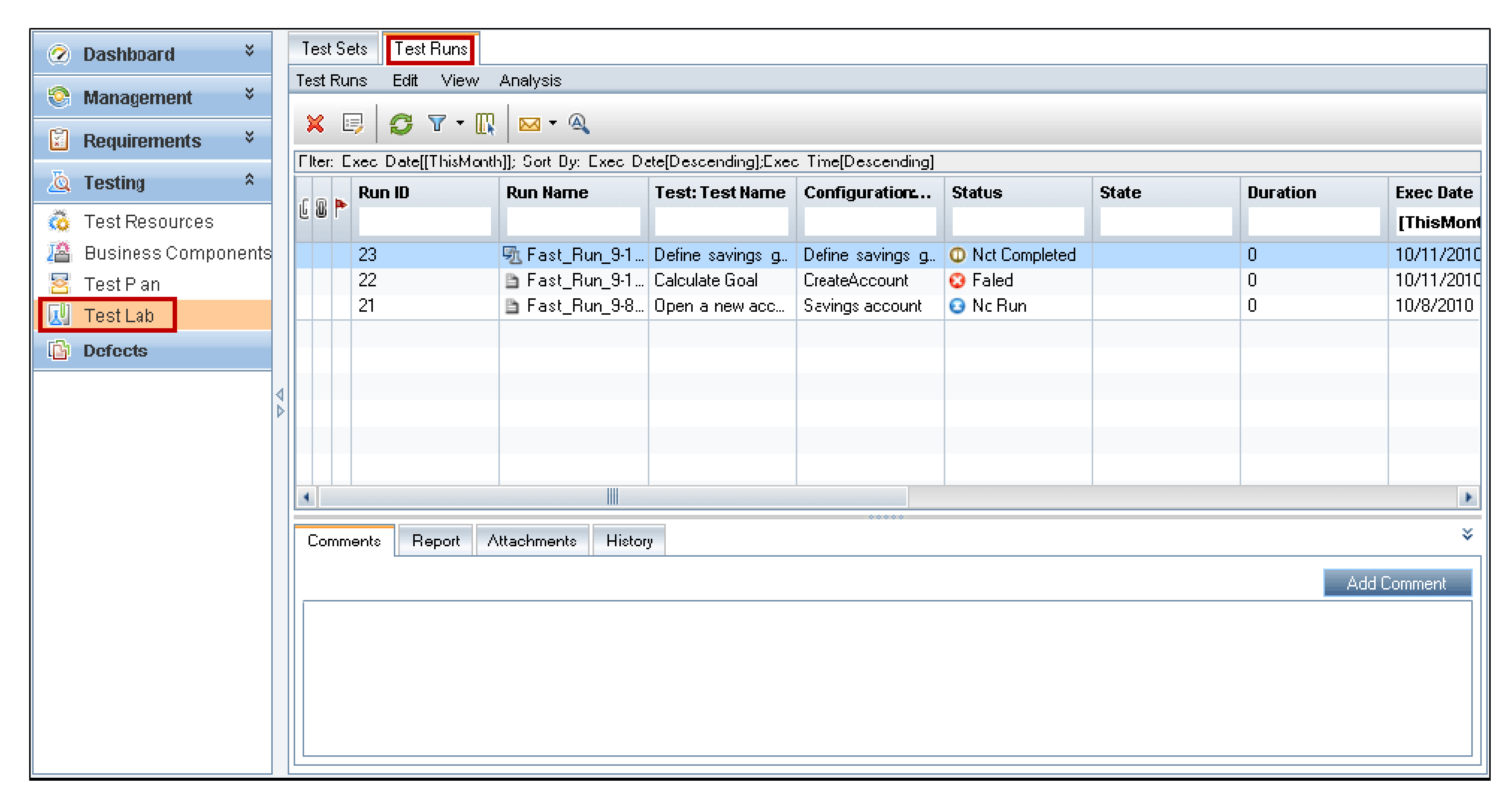


图 12-3　Test Runs 标签

12.4　测试执行概述

ALM 提供框架和工具以有效地执行测试(如图 12-4 所示)。其测试执行步骤如下。

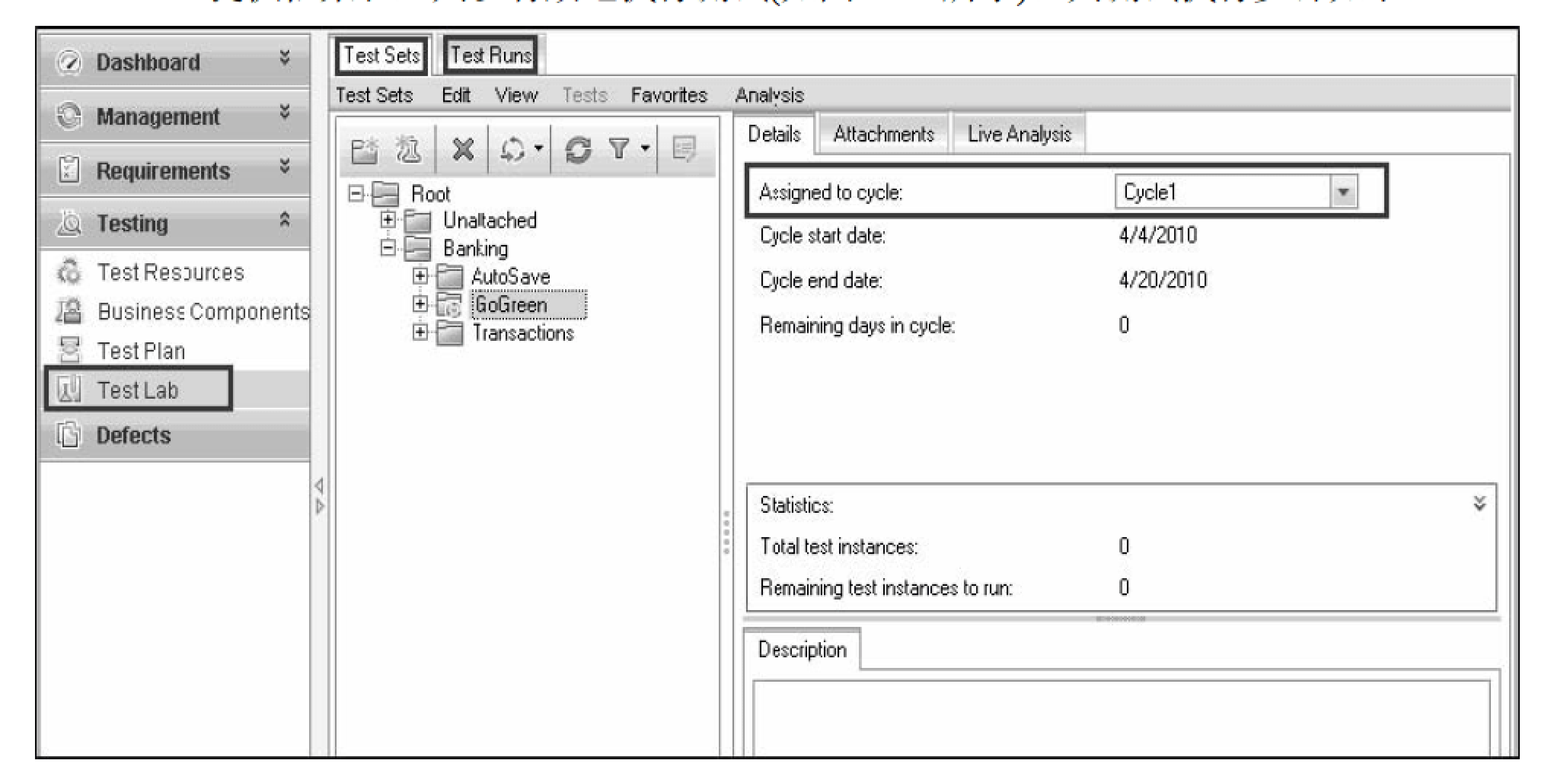


图 12-4　测试执行

1. 展开测试集树

提供测试构建模块的清晰图片，辅助实际测试的执行。能够帮助规划数据间的相互依赖关系，鉴别那些在未来测试中可能会被重用的常见脚本。创建测试集树：

1) 在测试集树中创建文件夹
2) 创建测试集
3) 向测试集添加测试

2. 组织测试运行

通过设置测试运行条件和创建时间表，控制测试的自动执行。这一自动化使得测试可以在无人、通宵或系统执行其他任务需求的时候进行。测试运行自动化也使得测试脚本更具有重用性也更容易维护，因此测试更加模块化的测试和按顺序安排它们成为可能。

1) 定义测试运行顺序和条件
2) 预定执行日期和时间

3. 设置自动化选项

在测试集执行完成后通知质量中心自动向作者发送状态警告电子邮件。此外，对于指定测试返回数字和清理自动测试失败任务，能够自定义规则。

4. 连接测试集文件夹与发布和周期

在 Management 模块中跟踪测试进程。

5. 运行测试

管理手动和自动测试的执行，在文档中记录结果。

12.5　测试集

测试集是用来完成特定测试任务的测试组。测试集包含具有一定关系的手动和自动测试。在同一测试集或不同测试集中可以多次添加一个测试，以便重用。

为理解测试集的概念，分析一下冒烟测试。冒烟测试是验证软件应用程序最重要功能的非彻底软件测试。冒烟测试不测试软件应用程序的出色细节。设想要利用冒烟测试来测试在线应用程序的登录功能。创建了包含验证测试功能测试的测试集。这个测试集中的测试能够验证在线应用程序登录需要使用的用户名和密码。

建立另一个包含测试的测试集来验证特定 Windows 环境中的登录功能。这个包含测试的测试集验证不同 Windows 系统、不同网页浏览器中的登录功能，如 Windows XP 和 Windows 2000 专业版。这两个测试集联合起来测试在线应用程序的所有方面。

12.5.1　测试集树

测试集树分层次组织并显示测试集。通过将测试集分组到文件夹，将文件夹按不同等级水平组织，开发一组测试集树有利于组织测试过程。

测试集树包含根级别的主要文件夹以指明测试集的大体分类。主文件夹包含每一级别中

进一步分类测试集的子文件夹。例如，图 12-5 显示了根目录下包含 Banking 文件夹的测试集树。这个文件夹中包含 Functionality and UI 和 Performance and Load 这两个按测试类型分类的测试集子文件夹。其中每个子文件夹又含有其下的测试集。

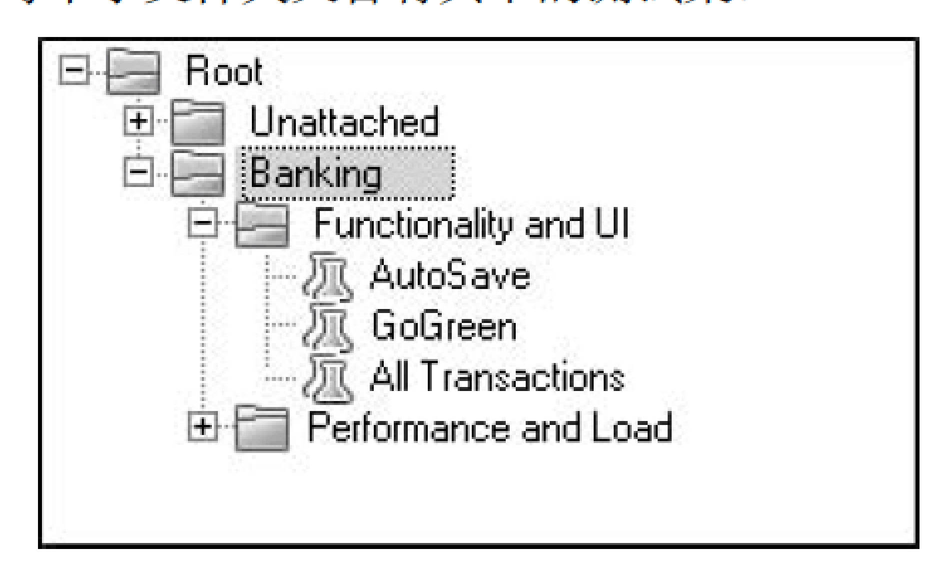


图 12-5　测试集树

12.5.2　创建测试集文件夹

测试集树总以 ROOT 文件夹开始。在这一文件夹中，可以创建主文件夹并向主文件夹中添加子文件夹(如图 12-6 所示)。添加文件夹的步骤如下。

1) 在测试集树中，选择 Root 文件夹创建主文件夹或选择存在的文件夹创建子文件夹。

2) 在工具条上，单击 New Folder 按钮。打开 New Test Set Folder 对话框。

3) 在 New Test Set Folder 对话框中的 Test Set Folder Name 区域中，输入新文件夹的名字。

注意：文件夹名字不能包含以下任何字符：\、^或 *。

4) 单击 OK 按钮向测试集树添加文件夹。

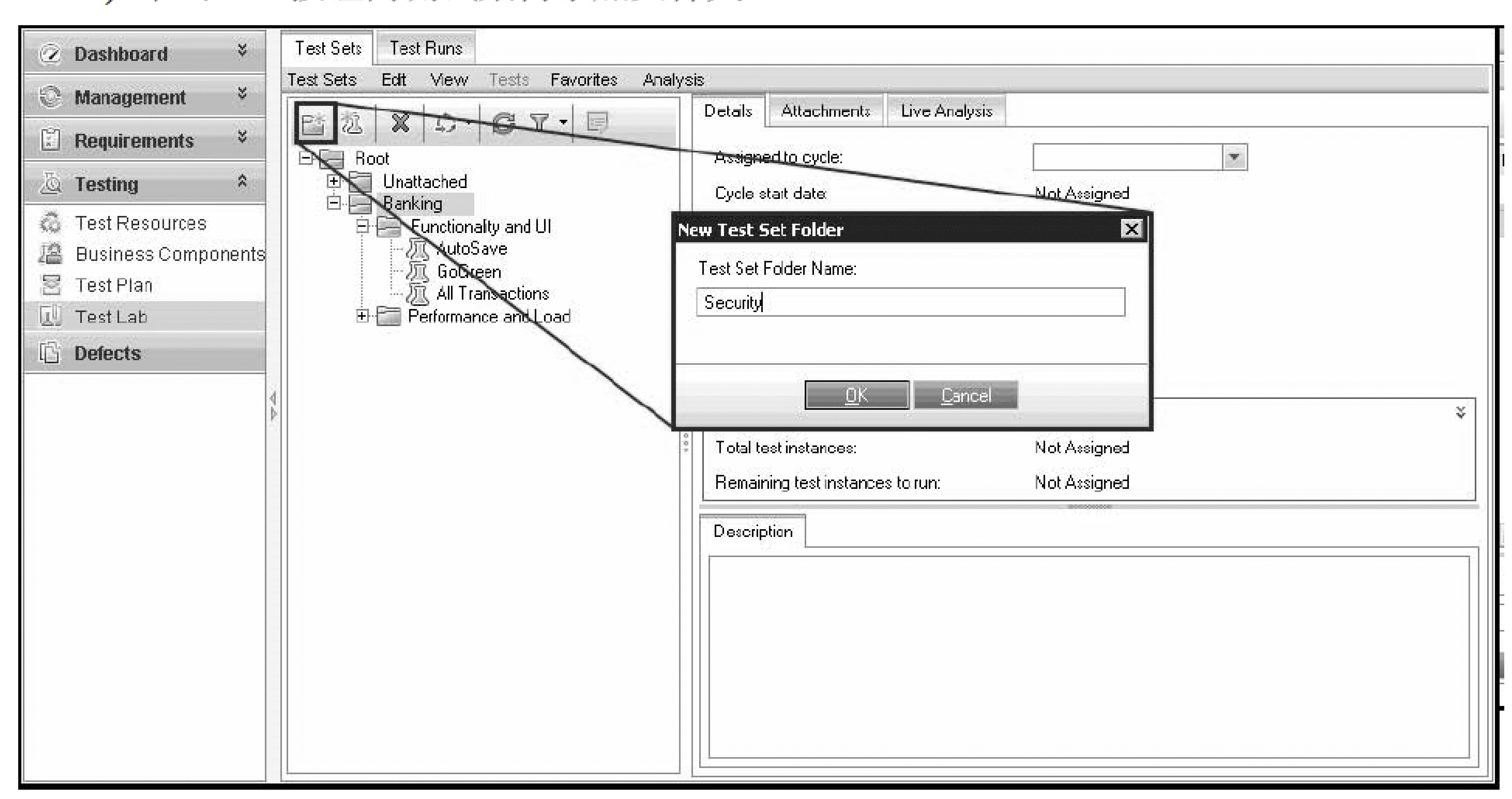


图 12-6　创建测试集文件夹

注意：文件夹包含子文件夹，每个子文件夹可以包含更深层的子文件夹。每个文件夹或子文件夹最多可以包含 676 个子文件夹。

12.5.3　创建测试集

创建完测试集文件夹后，在文件夹中创建测试集(如图 12-7 所示)。创建测试集的步骤如下。

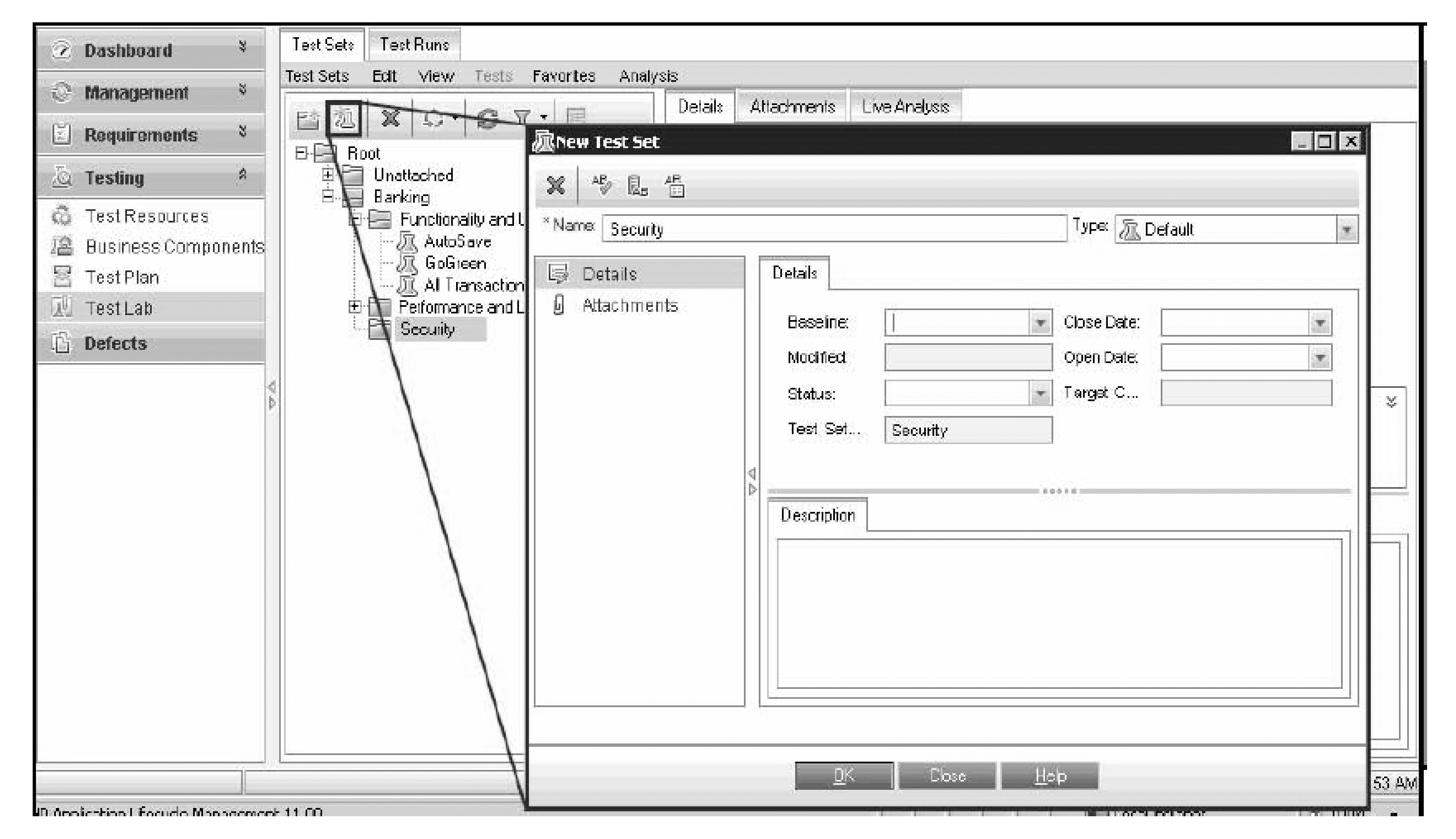


图 12-7　创建新测试集

1) 在测试集树中选择要添加新测试集的文件夹。

2) 在工具栏上，单击 New Test Set 按钮。弹出 New Test Set 对话框。

3) 在新测试集对话框中的测试集名称区域中，输入测试集名字。

4) 在 Description 字段，输入测试集的描述。

注意：测试集名称中不能包含下列任何字符：\、^ 或 *。

5) 单击 OK，关闭 New Test Set 对话框。新测试集出现在测试集树中。

12.5.4　调试测试集

ALM 的测试操作模块由以下测试执行元素组成，用来提供测试集信息(如图 12-8 所示)。

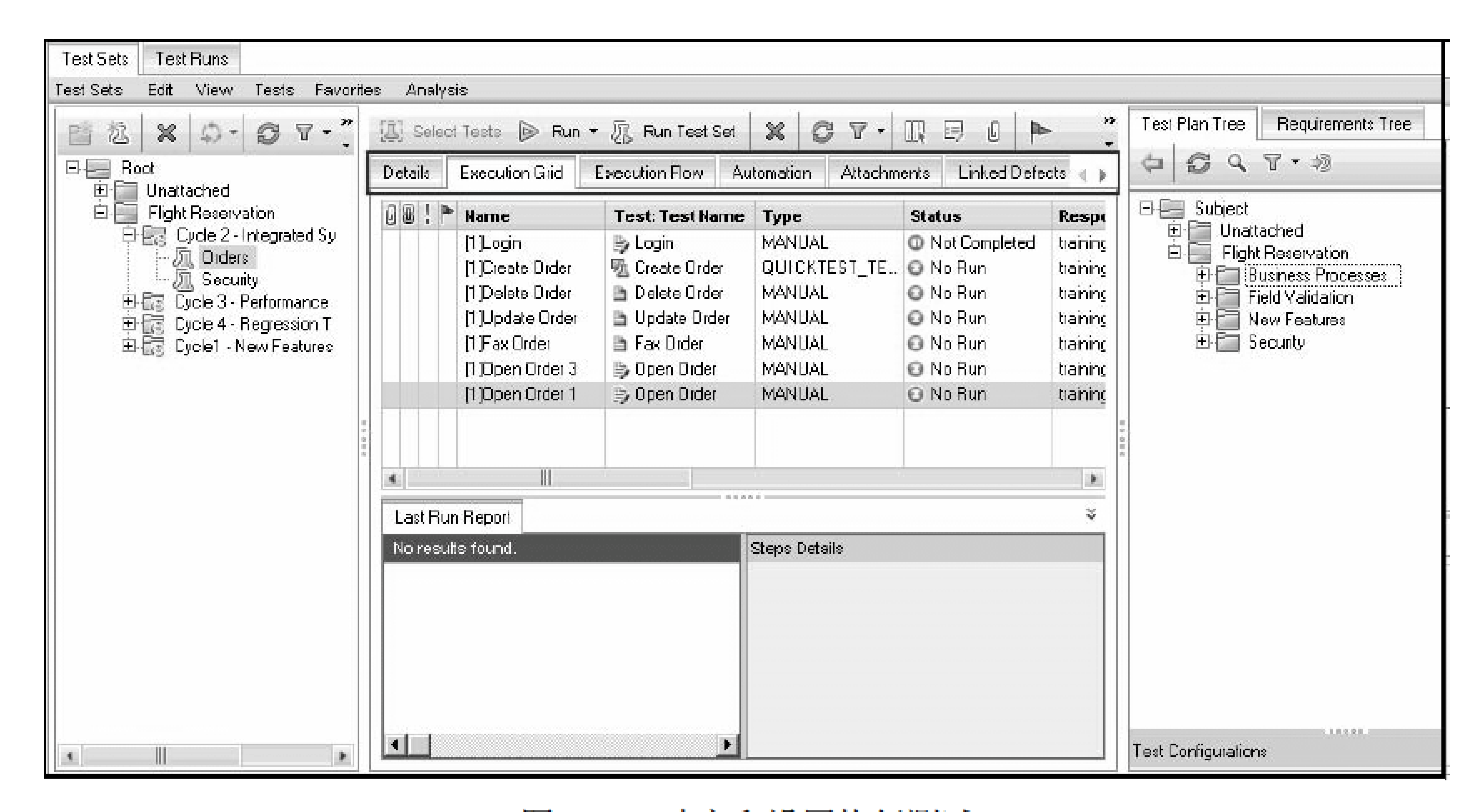


图 12-8　建立和设置执行测试

1) Details 标签：对当前测试集树中所选的测试集进行的描述。

2) Execution Grid 标签：声明组成测试集的测试、运行测试、检测执行结果、在网格中显示测试数据。

3) Execution Flow 标签：在图表中显示测试数据，为添加、排序和时间排序提供拖曳功能。

4) Automation 标签：显示邮件通知规则和当前选中测试集树中测试集失败的说明。

5) Attachments 标签：提供当前选中测试集树中测试集额外信息的附件列表。

6) Linked Defects 标签：查看与测试相关联的缺陷，也可以向测试添加新缺陷。

7) History 标签：查看当前选中对象的变化清单。此外，标签以对象显示为基准显示历史记录。

12.5.5　添加测试集细节

测试集创建后，向测试集中添加细节，例如其终止时间和其执行的测试周期(如图 12-9 所示)。为测试集提供额外信息的步骤如下。

1) 在测试集树中，选择一个测试集；

2) 在右边面板中，单击 Detail 标签；

3) 在 Detail 标签中，指定以下成员的值：(1)Close Date：显示计划的测试集关闭时间；(2)Baseline：在 Baseline 中，选择针对测试集的基准；(3)Open Date：显示计划的测试集起始时间；(4)Status：在 Status 中，设置测试集 Open 或 Closed 的状态。

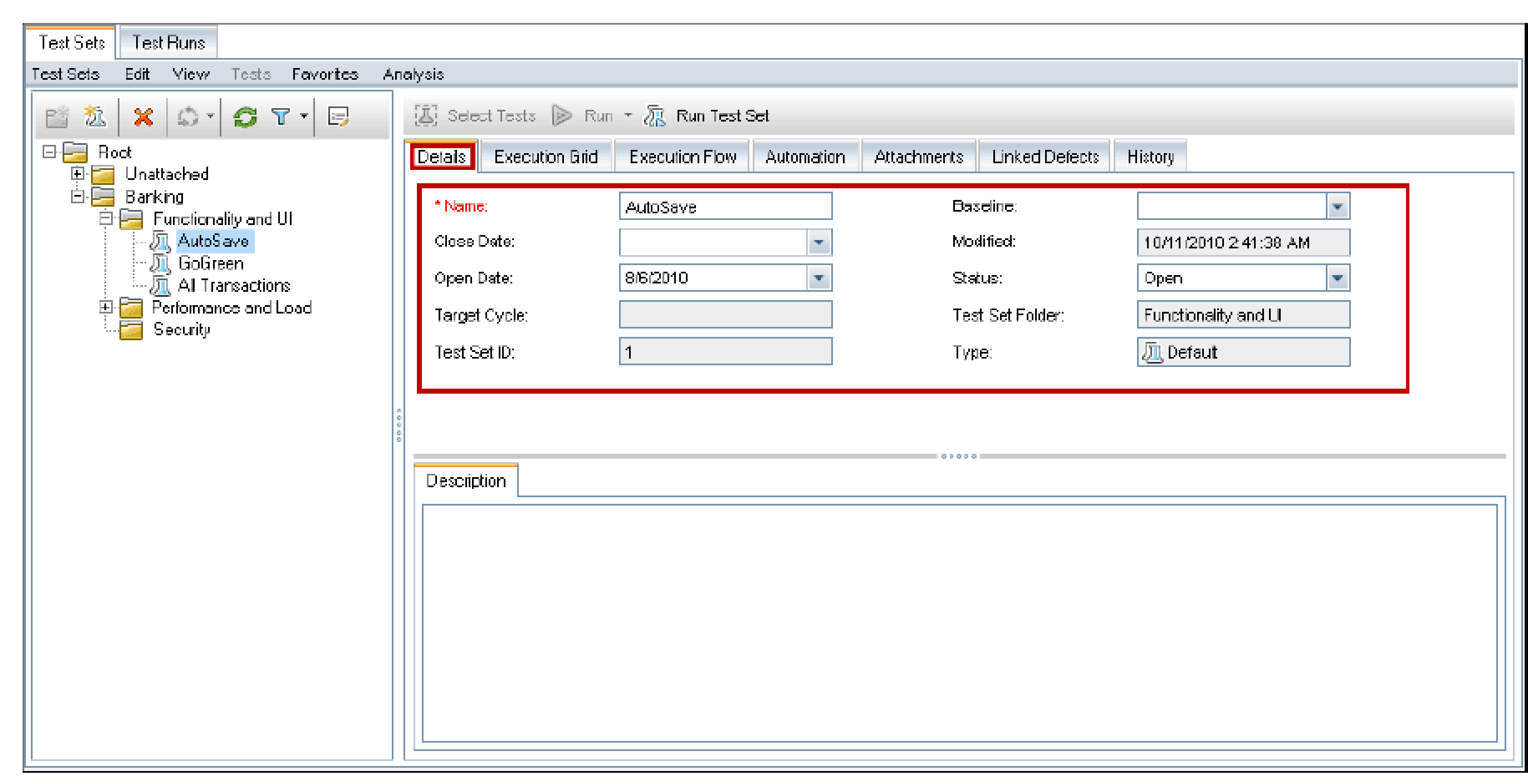


图 12-9　添加测试集细节

注意：向测试集添加附件，使用 Attachments 按钮。添加附件的过程和选项与 Requirements 模块中的一致。

12.5.6　为测试集添加测试配置

测试集包括任何或所有为测试定义的测试配置，也可以包括基于需求覆盖范围的测试配置。运行测试集时，根据每个测试配置定义的设置，对来自数据源的参数值进行恢复。图 12-10

显示为测试集添加测试配置的相关选项。

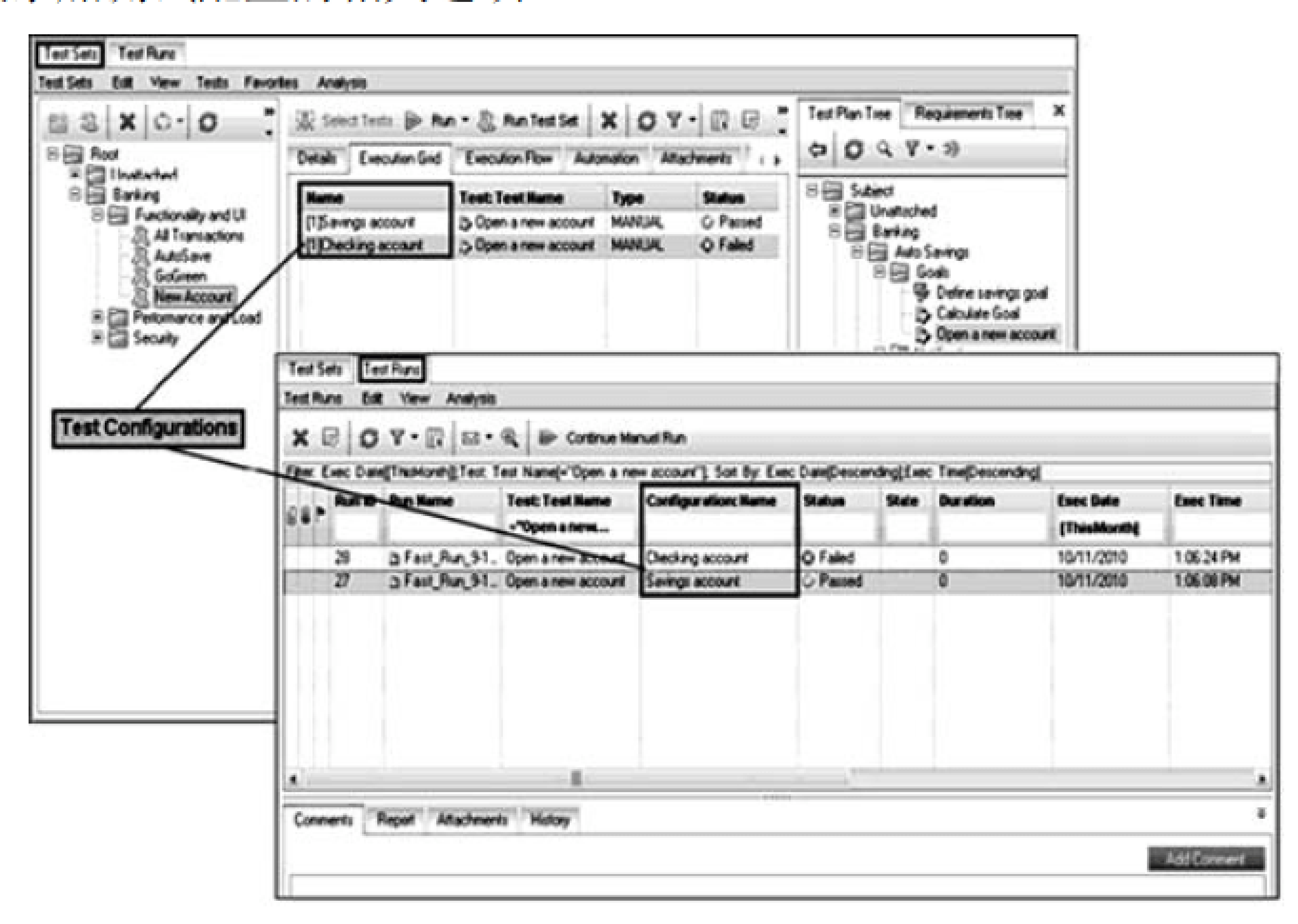


图 12-10　显示为测试集添加测试配置

1. 为测试集添加所有测试配置

1) 在测试计划树中选择测试，单击 Add Tests To Test Set 按钮。

2) 与测试相关的测试配置被添加到执行网格中。

2. 在测试运行后观察测试配置状态

执行网格中的测试运行后，单击 Test Runs 标签。Test Runs 标签中列出了测试配置和每个配置的状态。

12.5.7　基于需求覆盖范围的测试配置

添加基于需求覆盖范围的测试配置的步骤如下，可参考图 12-11。

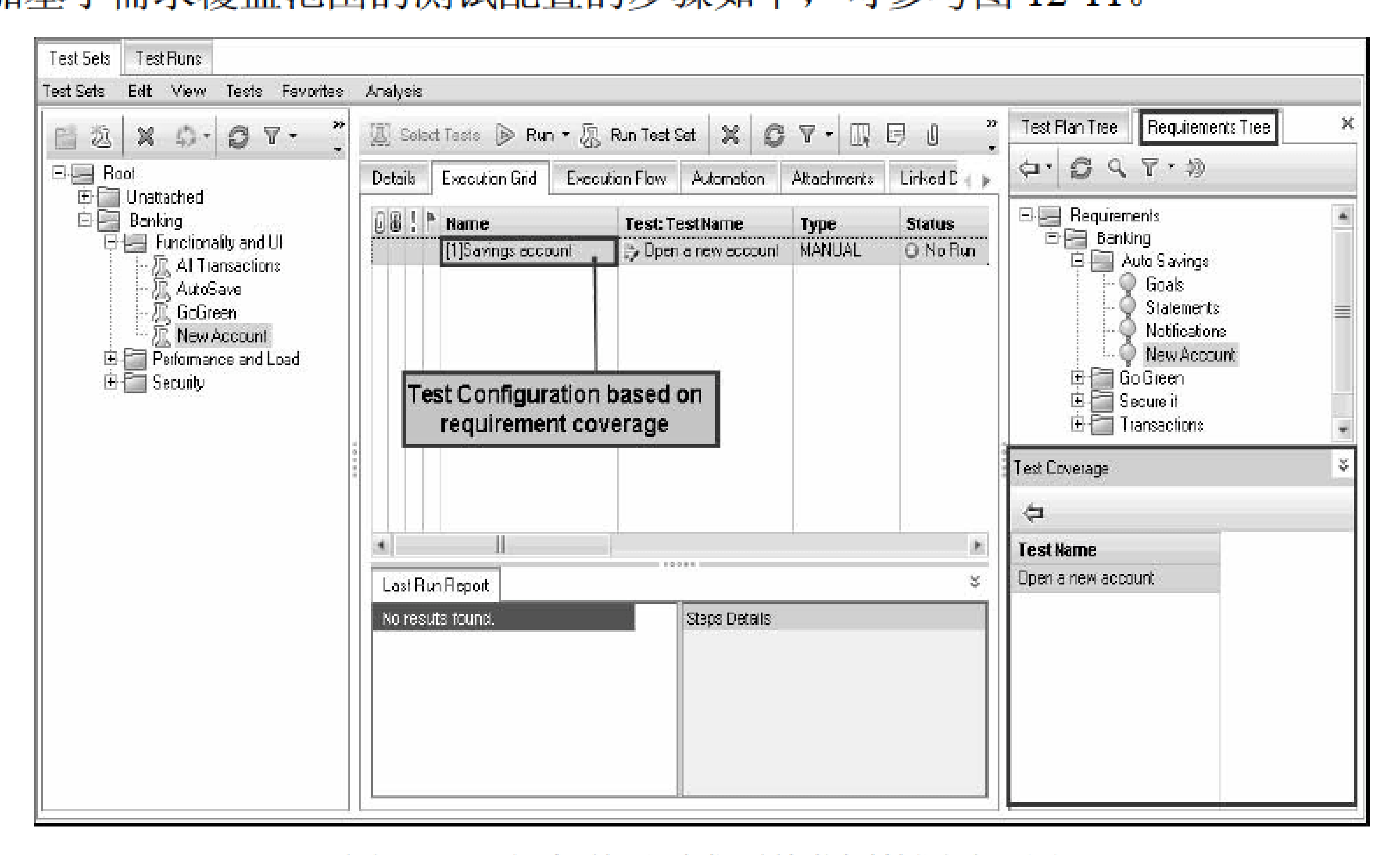


图 12-11　添加基于需求覆盖范围的测试配置

1) 单击 Requirements Tree 标签；

2) 在需求树中选择需求；

3) 展开 Test Coverage 面板。选择 Test Name，单击 Add Test 按钮。信息确认框弹出来询问是否确认添加；

4) 单击 Yes 按钮；

5) 与需求相关的测试配置被添加到执行网格中。

12.5.8　将测试集指定给周期

测试集文件夹中包含能够分配给管理模块周期的测试(如图 12-12 所示)。这一关联允许进行测试进程的检查，确定已解决的和突出的缺陷数量，增强报告能力。

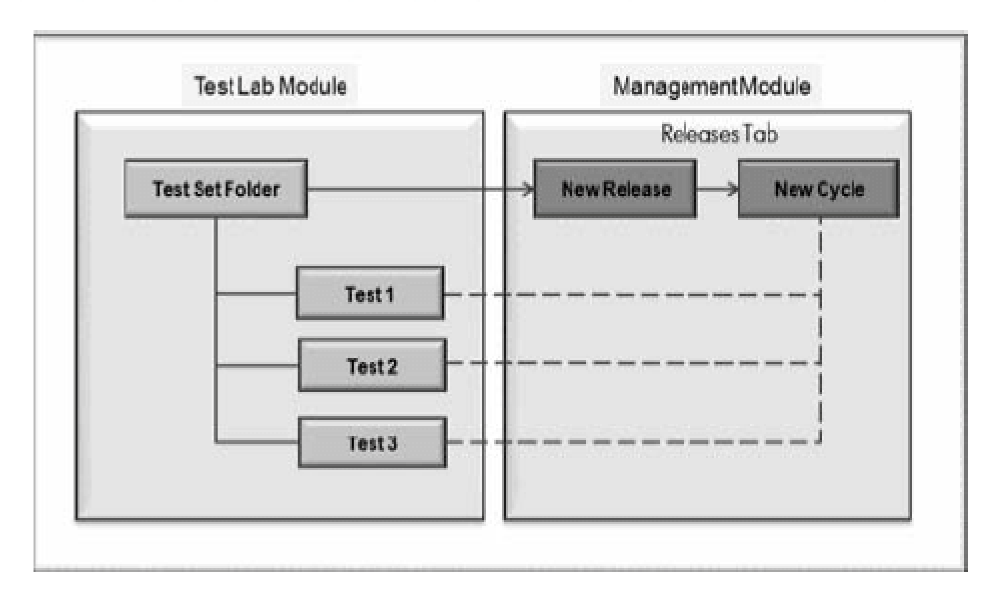


图 12-12　将测试集文件夹指定给周期

12.5.9　将测试集文件夹关联到周期

创建测试集树后，将测试集文件夹与 Management 模块中定义的周期相连(如图 12-13 所示)。当连接实现后，所有文件夹中的测试集在文件夹连接的周期中执行。

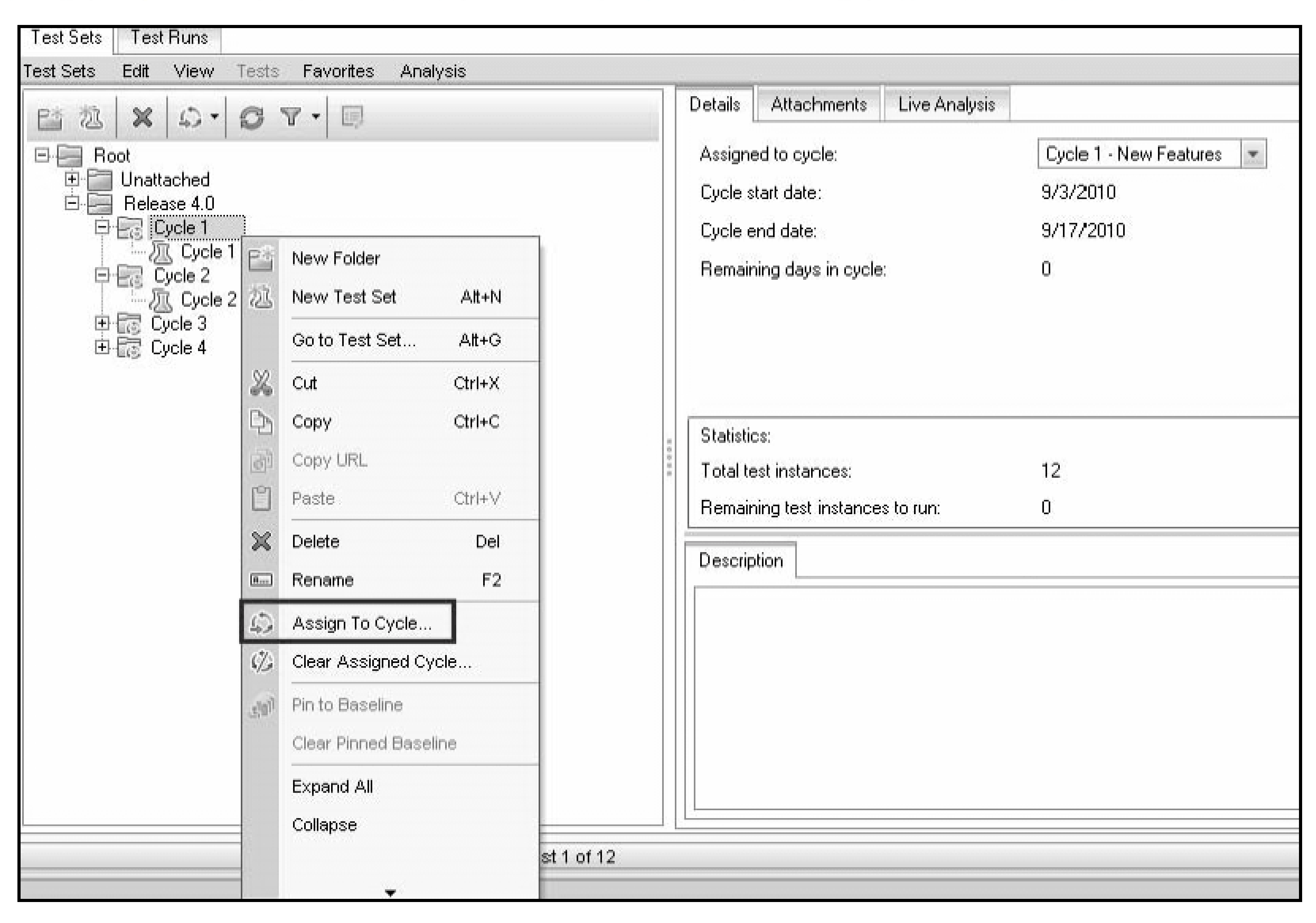


图 12-13　将测试集文件夹关联到周期

将测试集文件夹连接到发布周期的步骤如下。

1) 在测试集树中，右击测试集文件夹，选择 Assign To Cycle，弹出 Select Cycles 对话框。

2) 在选择周期对话框中，展开发布树。

3) 在发布树中选择一个周期。

4) 单击 OK，关闭 Select Cycles 对话框。

5) 测试集文件夹与 Select Cycles 对话框中的发布周期相连。

12.6　测试运行设置

12.6.1　使用 Execution Flow 标签

向测试集添加测试后，定义测试执行流来确定测试执行顺序。同样也要指定连续测试之间的从属关系。例如，可在测试通过前指定执行一个测试。在 Test Lab 模块，使用 Execution Flow 标签：

1) 生动地绘出每个测试集中测试的执行流；

2) 根据测试的执行完成状态安排测试运行。这些测试与其控制测试用实线连接。例如，在图 12-14 中，Autosave 和 Existing Account 测试之间用实现连接；

3) 为其他测试安排独立运行的测试。这些测试与测试部分之间用虚线连接；

4) 安排测试在指定的日期和时间运行。这些测试会显示一个时钟图标。例如，在图 12-14 中，Savings Account 测试显示一个时钟图标。时钟图标显示了执行测试的日期和时间。

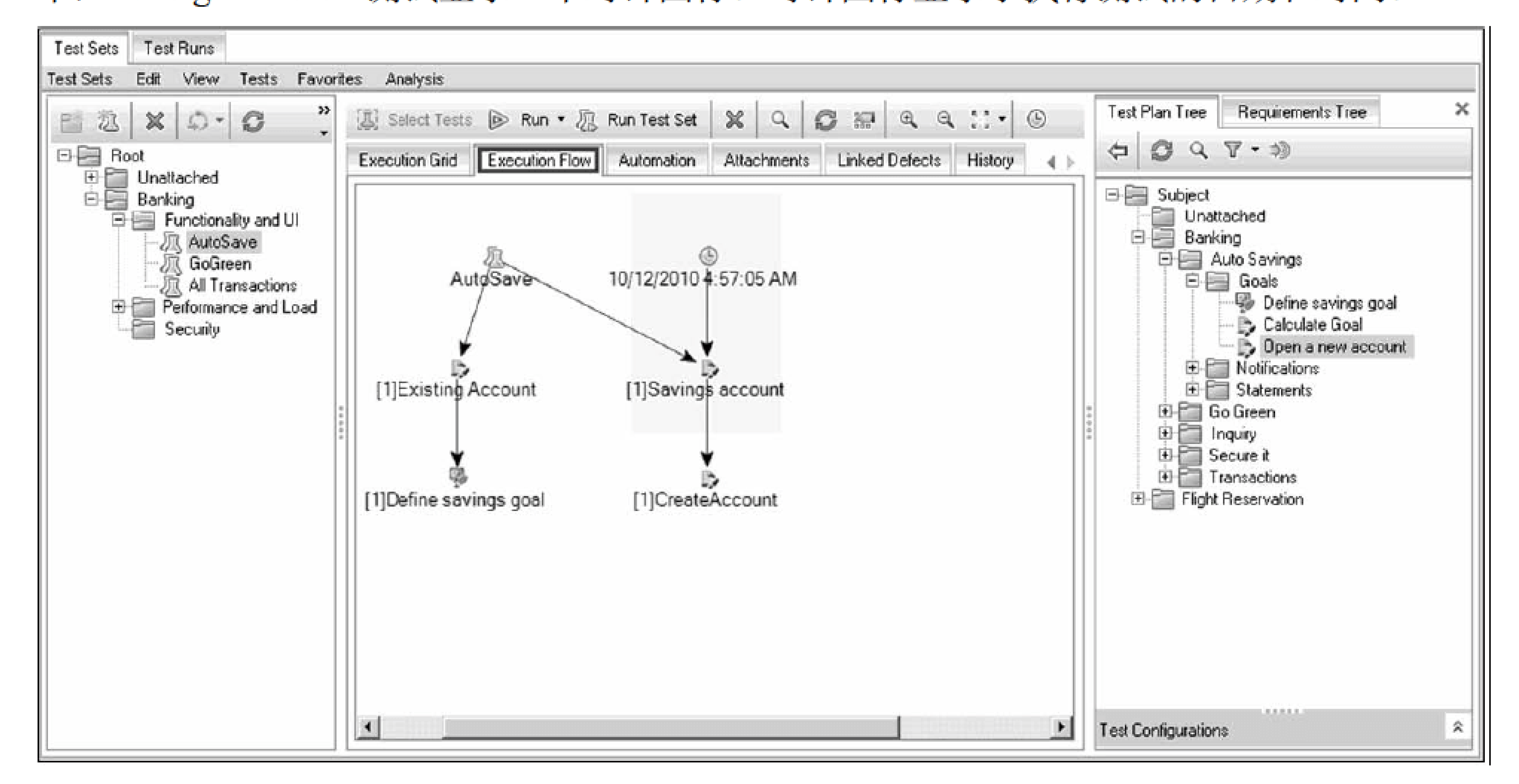


图 12-14　Execution Flow 标签

12.6.2　定义测试运行顺序和条件

通过定义描述指定测试要在其他测试进行之前完成或通过的条件，可建立测试执行之间的依赖关系(如图 12-15 所示)。定义测试执行条件的步骤如下。

1) 在测试集树中，选择测试集，单击 Execution Flow 标签。

2) 在 Execution Flow 页，双击一个测试。弹出 Run Schedule: Test 对话框，Execution Conditions 标签出现。

3) 添加条件，单击 New Execution Conditions。弹出 New Execution Conditions 对话框。

4) 在 New Execution Conditions 对话框中，在 Test 列表执行测试中选择测试。在下个域中，选择引发这个执行的参照测试的完成状态。

(1) 选择 FINISHED 设置规则，不管参照测试成功与否，依赖测试在参照测试执行之后执行。

(2) 选择通过 PASSED 设置规则，依赖测试只在参照测试执行成功之后执行。在 Execution Flow 标签中，有一条连接参照测试和测试的绿色实线。

5) 单击 OK，关闭 New Execution Condition 对话框。

6) 为同一测试设置其他条件，重复步骤 3 到 5。

7) 单击 OK，关闭 Run Schedule: Test 对话框。

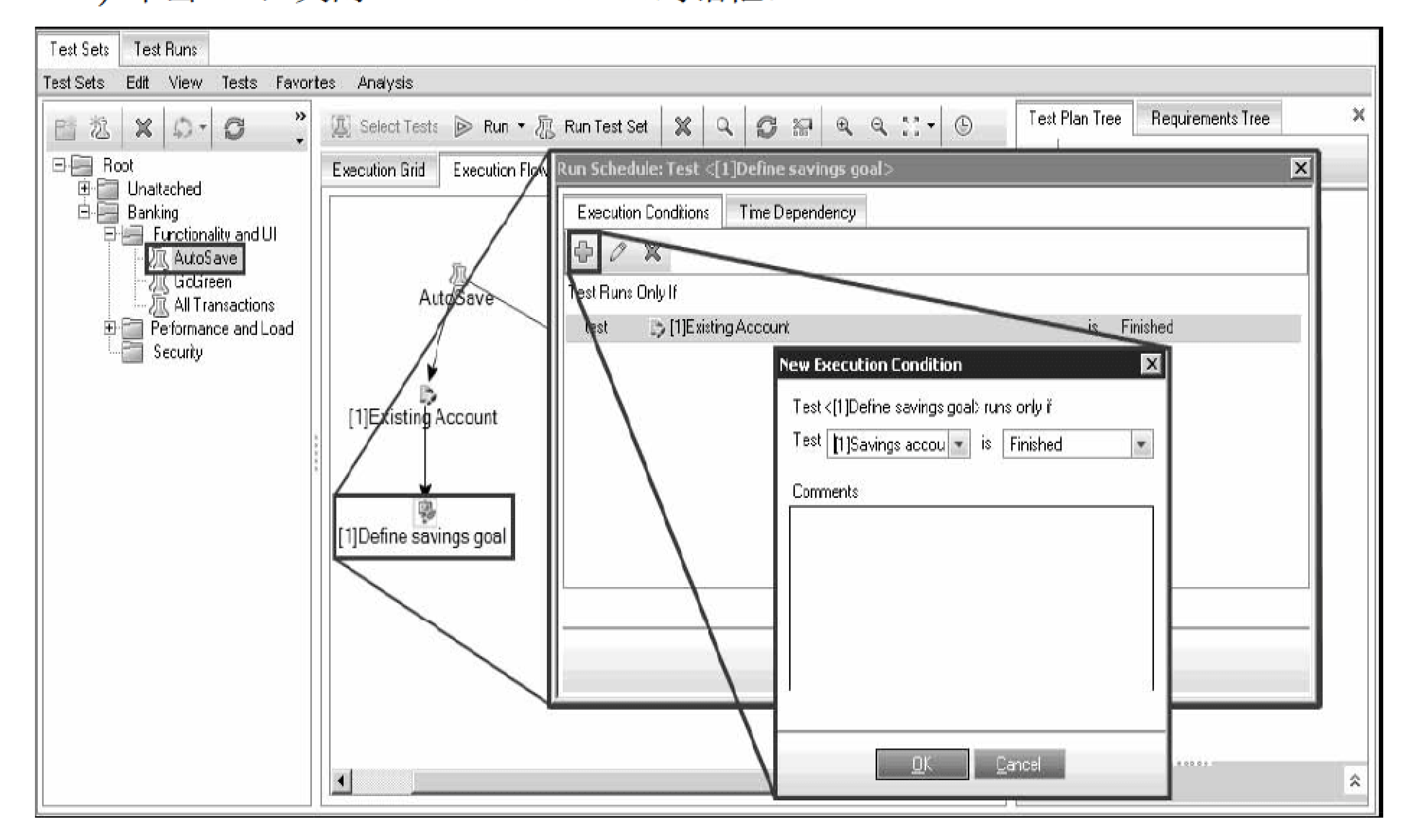


图 12-15　定义测试运行顺序和条件

12.6.3　安排执行日期和时间

除了定义测试运行顺序和条件，也可以指定测试执行的日期和时间(如图 12-16 所示)。指定测试执行的日期和时间的步骤如下。

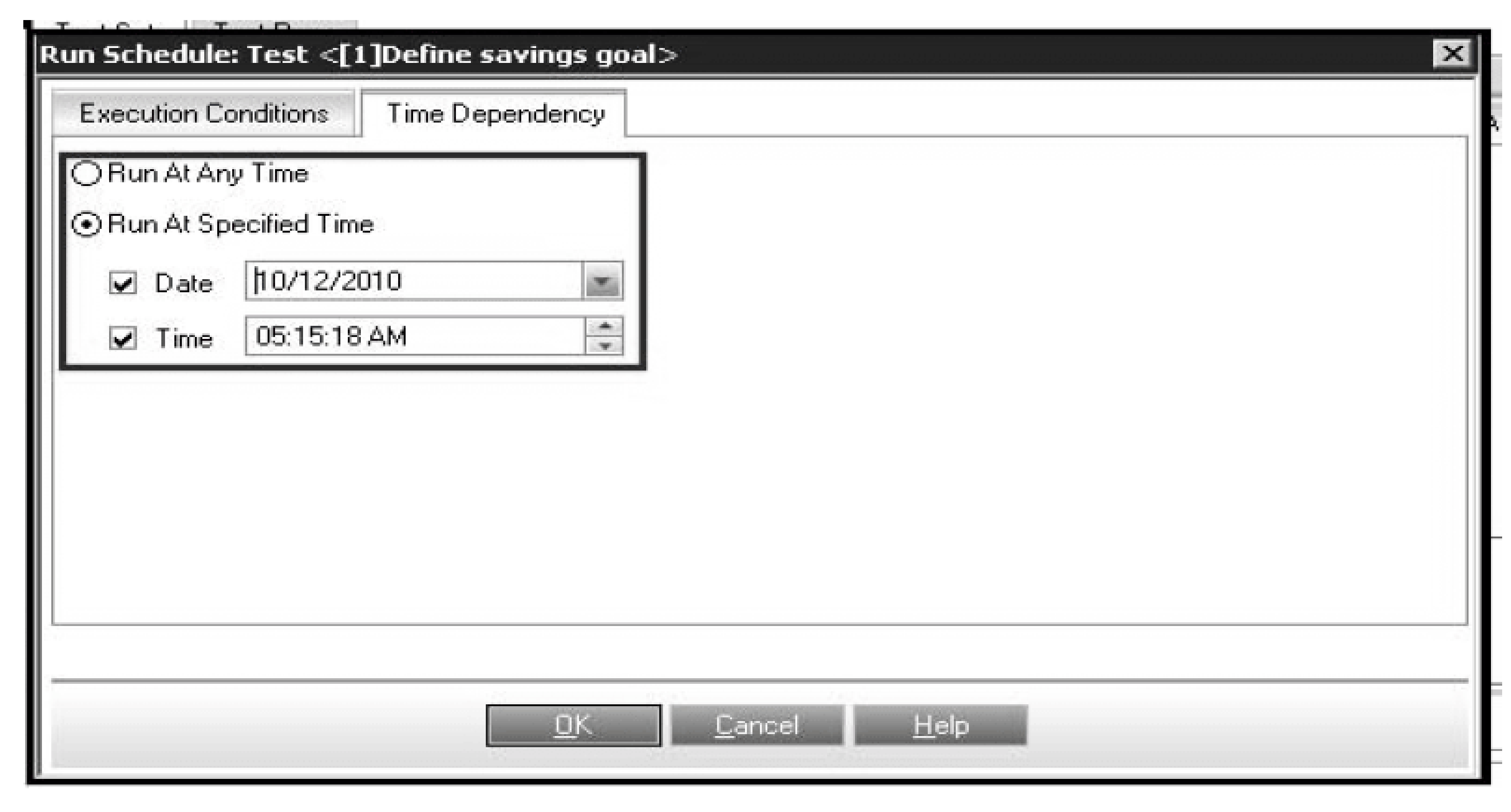


图 12-16　安排执行日期和时间

1) 在测试集树中，选择测试集，单击 Execution Flow 标签；

2) 在 Execution Flow 页面，双击测试。弹出 Run Schedule: Test 对话框；

3) 在 Run Schedule: Test 对话框中单击 Time Dependency 标签；

4) 选择 Run at Specified Time 选项；

5) 选中 Date 单选框，输入日期安排或在日历中选择日期；

6) 选中 Time 单选框，输入时间安排；

7) 单击 OK，关闭 Run Schedule: Test 对话框。带时钟图标的测试出现在 Execution Flow 中。图标显示测试执行的日期和时间。

12.6.4　安排的附加选项

定义测试执行顺序，可使用以下两个方法中的一种：

1) 选中 CTRL，单击 Execution Flow 页面中的测试。然后在菜单条中选择 TESTS |Order Test Instances 命令。弹出 The Order Test Instances 对话框。使用对话框中的上下箭头按钮重新排列测试执行的顺序。

2) 选择图标，将参照测试拖曳到其依赖测试上，设置依赖测试的实现 Finished 条件。

注意：默认情况下，两个操作都会在对照测试和依赖测试之间连接一条蓝色实线。

3) 设置测试条件，双击对照测试和依赖测试之间的线。弹出执行条件对话框。选择完成或通过，指出依赖测试执行前的对照测试的状态。

4) 设置日期和时间顺序，单击 Add Time Dependency to Flow。在 Execution Flow 页面中会加入一个时钟图标。将时钟图标拖曳到需要安排的测试图标。可以看到测试和时钟图标之间的蓝线。双击蓝线打开 Time Dependent of Test 对话框。设置执行指定的日期和时间。

12.6.5　知识检查

如果知道测试集中的一个测试可能会失败，但要确保测试集中后续的测试能继续执行，应该考虑以下事项：

1) 断开测试与流的连接。

2) 执行条件的箭头指向设置为完成。

12.6.6　设置失败规则

每个测试集包含了定义测试需要返回多少次、当自动测试失败时应当进行清理的规则(如图 12-17 所示)。设置失败规则：

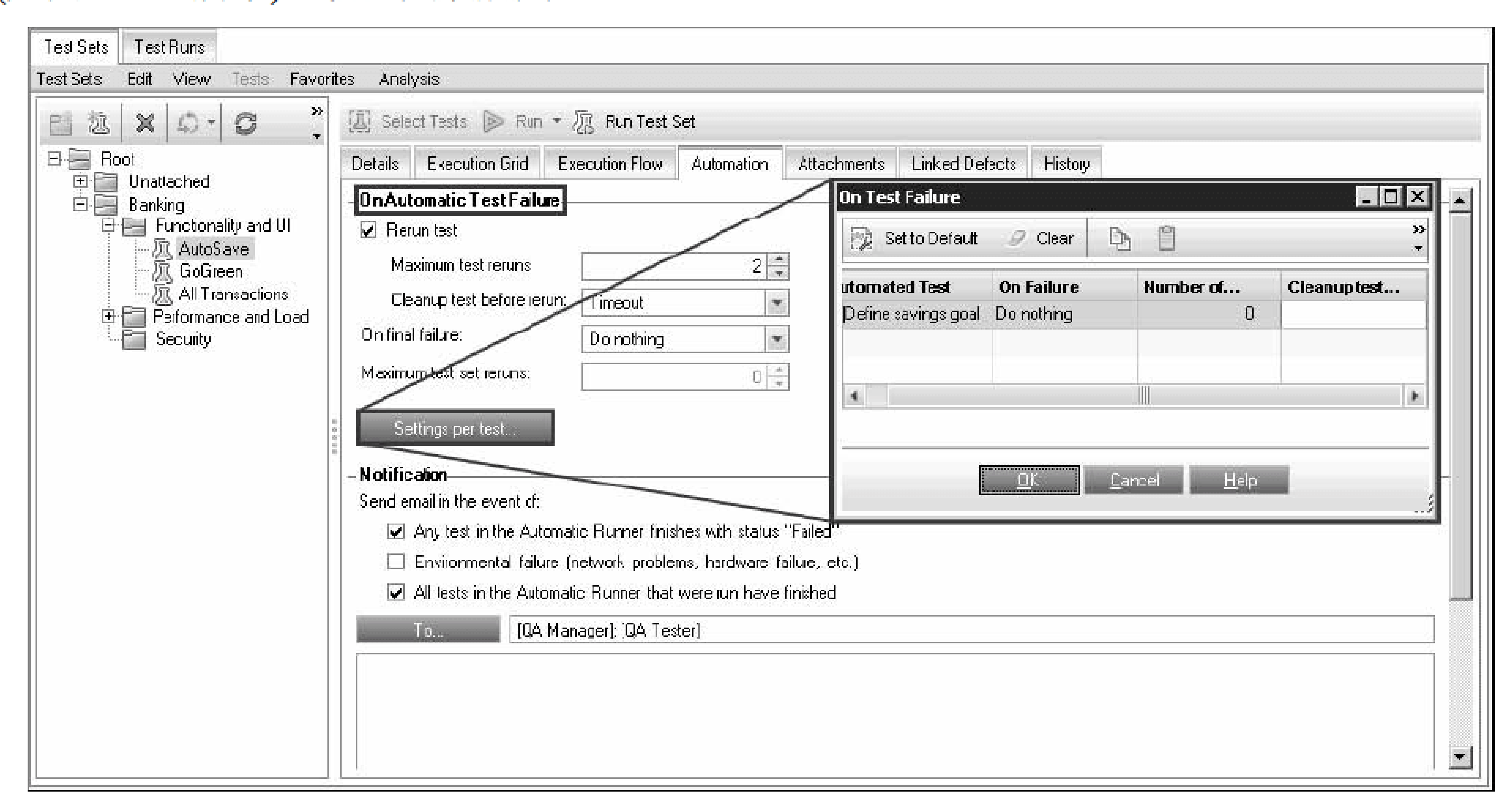


图 12-17　设置失败规则

1) 在测试集树中，选择测试集

2) 在右面板中，单击 Automation 标签

3) 在 OnAutomatic Test Failure 内容中，使用以下选项设置测试规则：

(1) 为测试集中的所有自动测试设置通用和默认失败规则：首先单击 Rerun test 单选框，在 Maximum test rerun 中，指定每个测试应当被重新运行的次数；然后单击 Cleanup test before rerun 按钮，选择所有被选测试中的通用清理测试。

注意：浏览按钮使得能够从测试计划树中选择清理测试。

(2) 为测试集中的每个自动化测试设置不同失败规则：(1)单击 Setting Per Test。弹出 On Test Failure 对话框，列举出测试集中所有的自动化测试；(2)为每个列入的测试设置 Rerun 次数；(3)为每个列入的测试设置 Cleanup Test，单击 OK。

4) 为定义任何测试在 On Failure 下发生的事，选择 DO Nothing、Stop the Test Set 或者 Rerun The Test Set。

12.6.7　设置测试集提示

通过配置测试集，在测试执行完成后可以自动地向测试集作者发送状态警告邮件(如图 12-18 所示)。发送状态警告邮件通知的步骤如下。

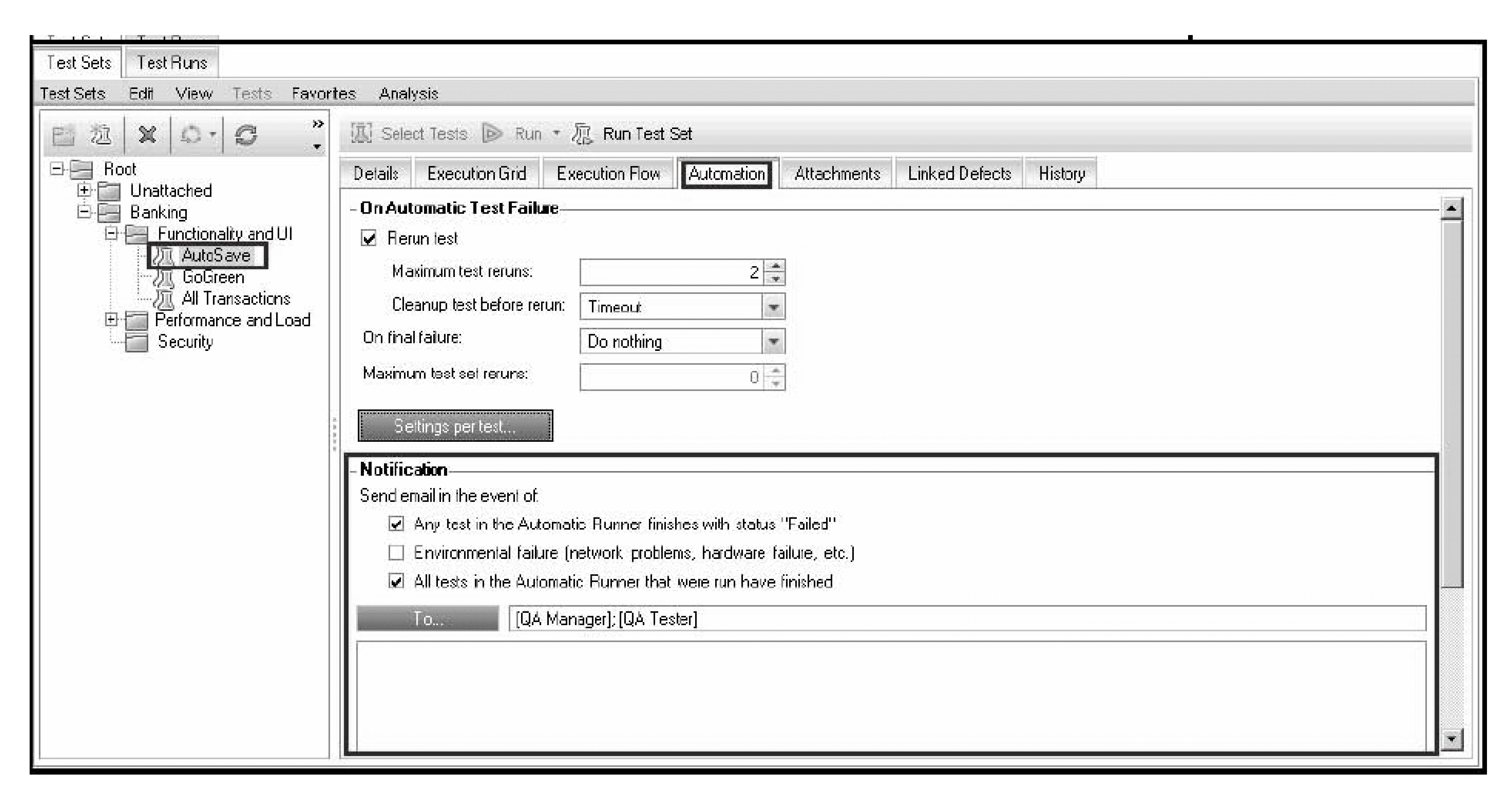


图 12-18　设置测试集提示

1) 在测试集树中，选择测试集。

2) 在右侧面板，单击 Automation 标签。

3) 发送通知要在 Notification 部分，选择邮件要发送的事件。

注意：如果测试集因为非测试逻辑原因失败，可通过选择 Environmental Failure 事件发送邮件。环境失败包括没有返回结果的函数调用、非法访问、应用程序组件之间版本不兼容、缺少 DLL 文件或没有足够授权。

1) 指定邮件接收者，键入其有效的邮件地址，或通过 To 按钮选择用户名或用户组名。

2) 在 Message 字段下，键入邮件消息的正文。

12.7　运行测试

手动或自动运行 ALM 测试和测试集的步骤如下。

1. 手动运行测试

在 ALM 中可以手动地运行手动和自动测试。手动运行测试(如图 12-19 所示)时，按照测试程序的步骤和执行操作进行。每一步的成败取决于实际程序结果和期望输出是否匹配。在 ALM 中可以手动地使用以下工具运行测试：(1)HP Sprinter，Sprinter 在手动测试过程中提供强化功能辅助；(2)Manual Runner。如果不适用 Sprinter，可使用 Manual Runner 进行手动测试。



图 12-19　手动运行测试

2. 自动运行测试

自动运行自动测试时，ALM 自动打开选择的测试工具，在本机或远程主机上运行测试，向 ALM 输出结果。

1) 使用 Automatic Runner 自动运行测试。

2) 可自动地运行手动或自动测试。

3. 在测试运行中编辑测试步骤

用 Manual Runner 执行测试，可以添加、删除、或修改测试步骤。测试执行完成后，可以保存修改过的设计步骤。使用 Manual Runner 在手动运行时编辑“测试步骤：步骤详细页”。

4. 重新启动手动测试的运行

如果手动测试在执行期间中止，可以继续运行后一步。继续运行与测试首次运行使用的是同一个运行器(Sprinter 或者 Manual Runner)。使用以下任意一种：

1) 在 Execution Grid 标签或 Execution Flow 标签中，选择想要重新运行的测试，选择 Tests | Continue Manual Run。另一种是单击 Run 箭头，选择 Continue Manual Run。

注意：如果要重新运行先前的运行，单击 Test Instance Details 按钮，在工具条上单击 Run，选择要重新启动的测试运行。

2) 单击 Continue Manual Run 按钮。在 Test Lab | Test Run 标签中，选择要重新启动的测试运行，单击 Continue Manual Run 按钮。

12.7.1　使用 Sprinter 运行手动测试

使用惠普 Sprinter 根据惠普生命周期管理(ALM)手动地运行测试。Sprinter 提供了高级的功能和工具帮助进行手动测试进程。如图 12-20 和图 12-21 所示。

手动运行并记录测试结果的步骤如下。

1) 在测试集树中选择测试集。

2) 在右侧面板中，单击 Execution Grid 或者 Execution Flow 标签，选择手动测试。

3) 在 ALM 工具条上，单击 Run 箭头，选择 Run Manually。弹出 Manual Runner: Test Set 对话框。

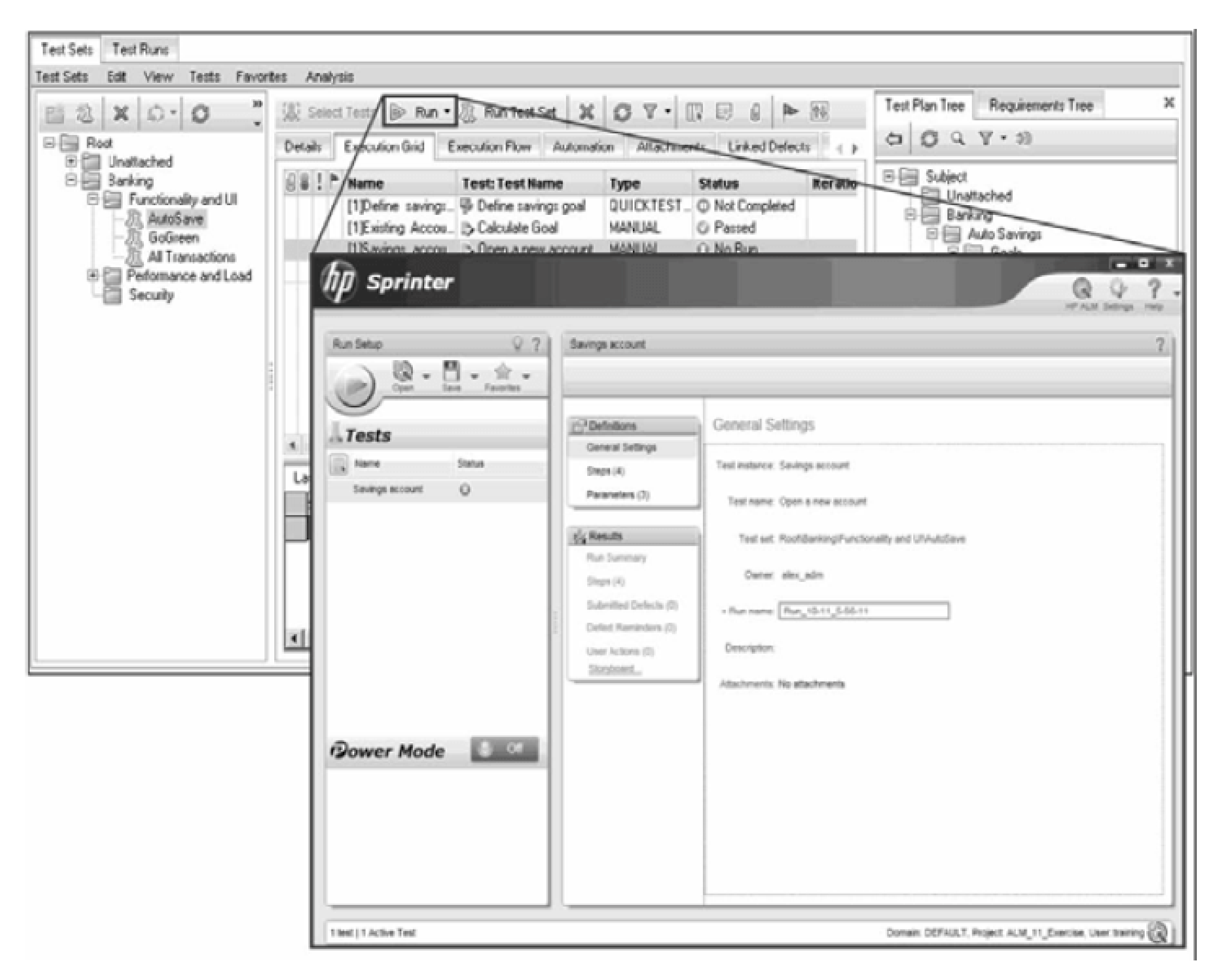


图 12-20　Sprinter 手动运行测试

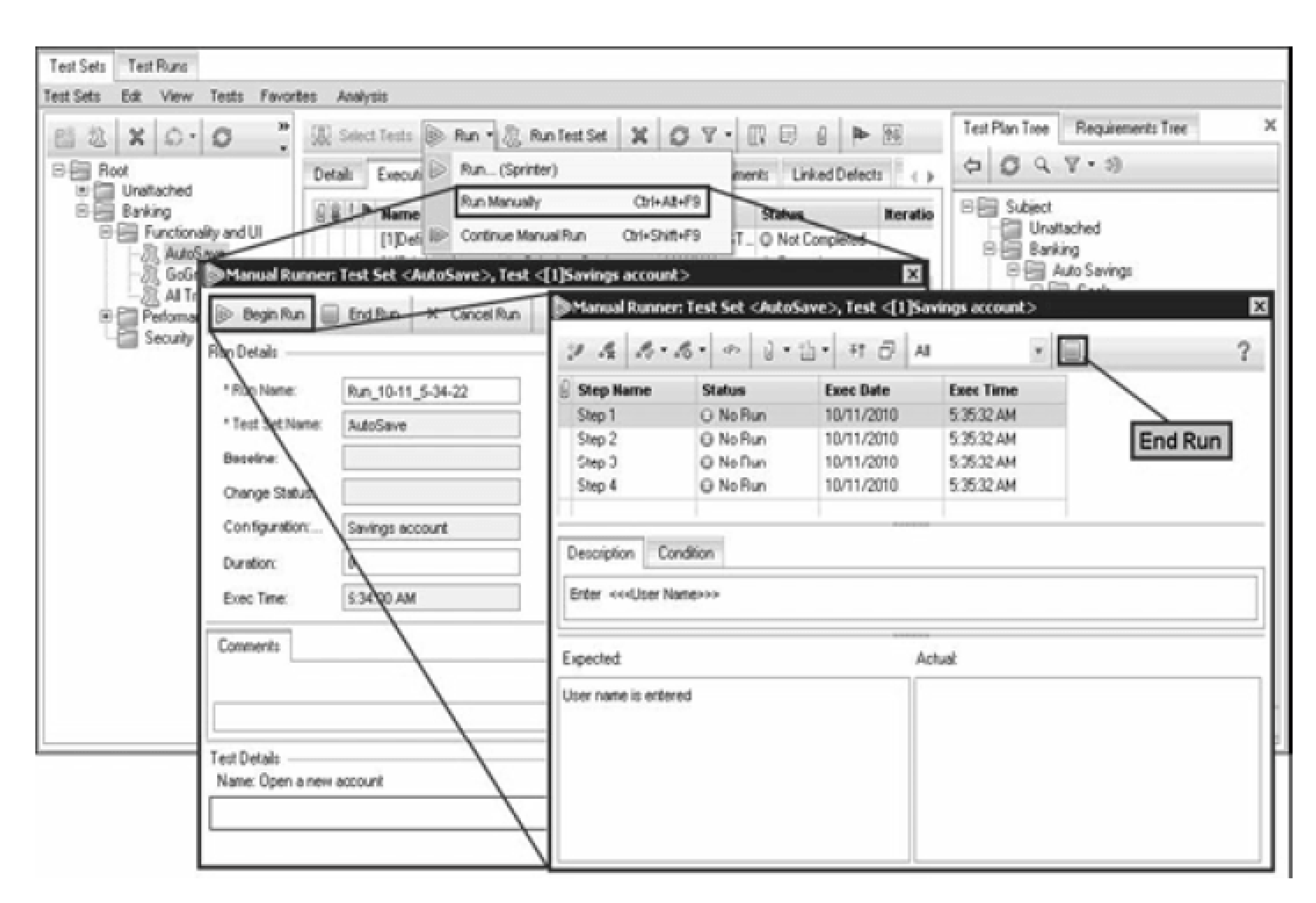


图 12-21　手动运行测试

4) 启动测试运行，单击 Begin Run。弹出 Manual Runner: Test Set 对话框。

注意：如果测试中有未设置的参数，在 Parameters Values For Run 对话框中会提示为未设置参数分配值。

1) 按照 Manual Runner: Test Set 对话框 Description 域中概述实现测试步骤。

2) 使用提供的字段记录每一步骤的状态和实际结果。

3) 结束测试运行，单击 End Run。

注意：运行状态为 Not Completed 的测试，在 ALM 工具条中单击 Run 箭头，选择 Continue Manual Run。

运行手动测试时，ALM 提供了记录手动测试结果的不同选项。

使用下面的选项记录手动测试运行结果，可参考图 12-22。

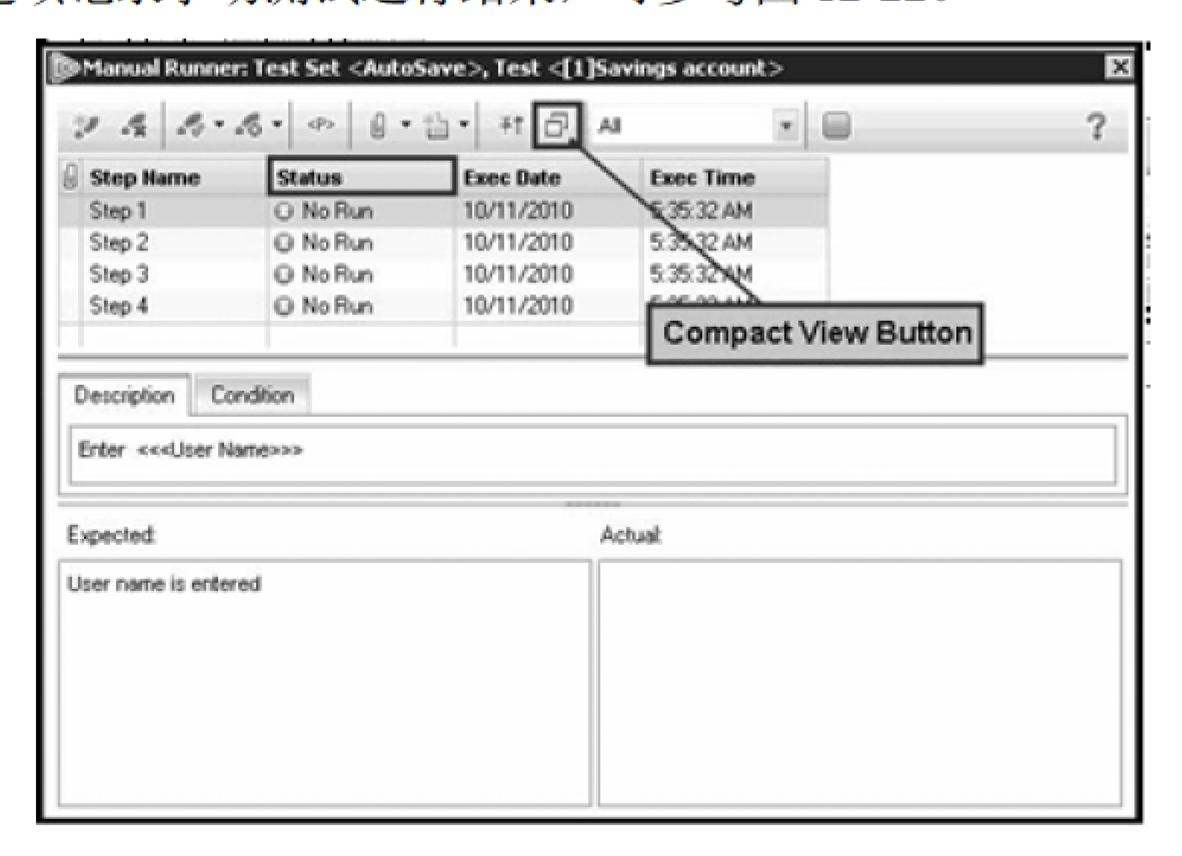


图 12-22　记录手动测试结果

1) Compact View 按钮：连接 Steps Grid 和 Compact View。使用 Compact View 分别观察、升级 Description、Exception、Actual 等测试步骤字段。

2) STATUS 列：记录测试的执行状态。

3) ACTUAL 字段：记录实际测试执行结果的额外细节信息。

12.7.2　自动化测试

使用 ALM 执行自动化测试有三点须知：

1) 使用自动测试多次执行测试用例。

2) ALM 运用多种自动工具创建和执行测试用例，例如 Quick Test Professional 和 LoadRunner。

3) 运行自动测试时，ALM 调用合适的测试工具，在指定主机上运行测试，获取测试执行结果。

自动运行测试设置的步骤如下，可参考图 12-23。

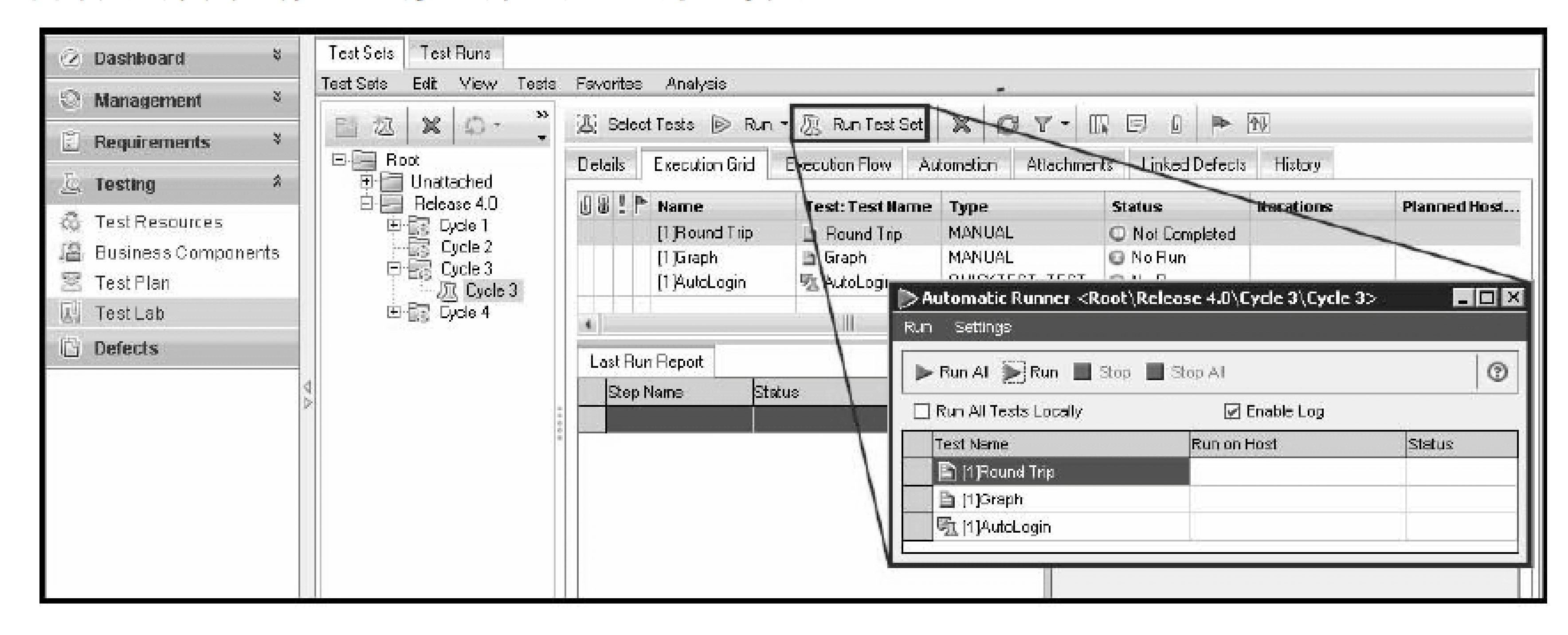


图 12-23　自动化测试设置

1) 在测试集树中选择测试集。

2) 在右面板，单击 Execution Grid 或 Execution Flow 标签，选择自动测试。

3) 单击 Run Test Set。弹出 Automatic Runner 对话框，显示所选测试。

4) 指定运行测试位置：(1)本地执行测试，选中 Run All Tests Locally 单选框；(2)远程执行测试，不选中 Run All Tests Locally 单选框。在 Run on Host 列，选择主机或主机组名。这一列提供了打开 Select Host 对话框的浏览按钮。

注意：如果 Select Host 对话框没有列举出需要的主机电脑，使用 Test Sets|Hostsmanager 菜单指令更新列表。

5) 启动测试执行，在 Automatic Runner 对话框中选择第一个测试，单击运行。

6) 终止测试运行，单击 Stop。

下面是对运行自动测试的重要指导原则：

1) 在远程机组设置测试，确保测试运行仅发生在组中第一个可行主机，而不是主机组中的任意主机。

2) 在多个远程主机上运行当前测试，在测试集中添加多个测试实例。然后，在 Automatic Runner 对话框，在每个测试实例的 Run on Host 列选择不同主机。

3) 在多个远程主机中运行当前同一个测试集，为每个测试集设置并执行单独的行测试运行。完成这些需要在每个测试集中打开 Automatic Runner 对话框。在 Automatic Runner 对话框中能设置和执行单独的测试运行。

4) 在远程主机上运行测试之前，需要首先配置 Host Manager 对话框。

12.8　主机管理器

配置远程测试执行主机的步骤如下。

任意连接到网络的主机都可运行测试。使用 Host Manager 对话框，可以为测试执行的可行主机创建列表，也可以为用于特定项目的主机分组(如图 12-24 所示)。

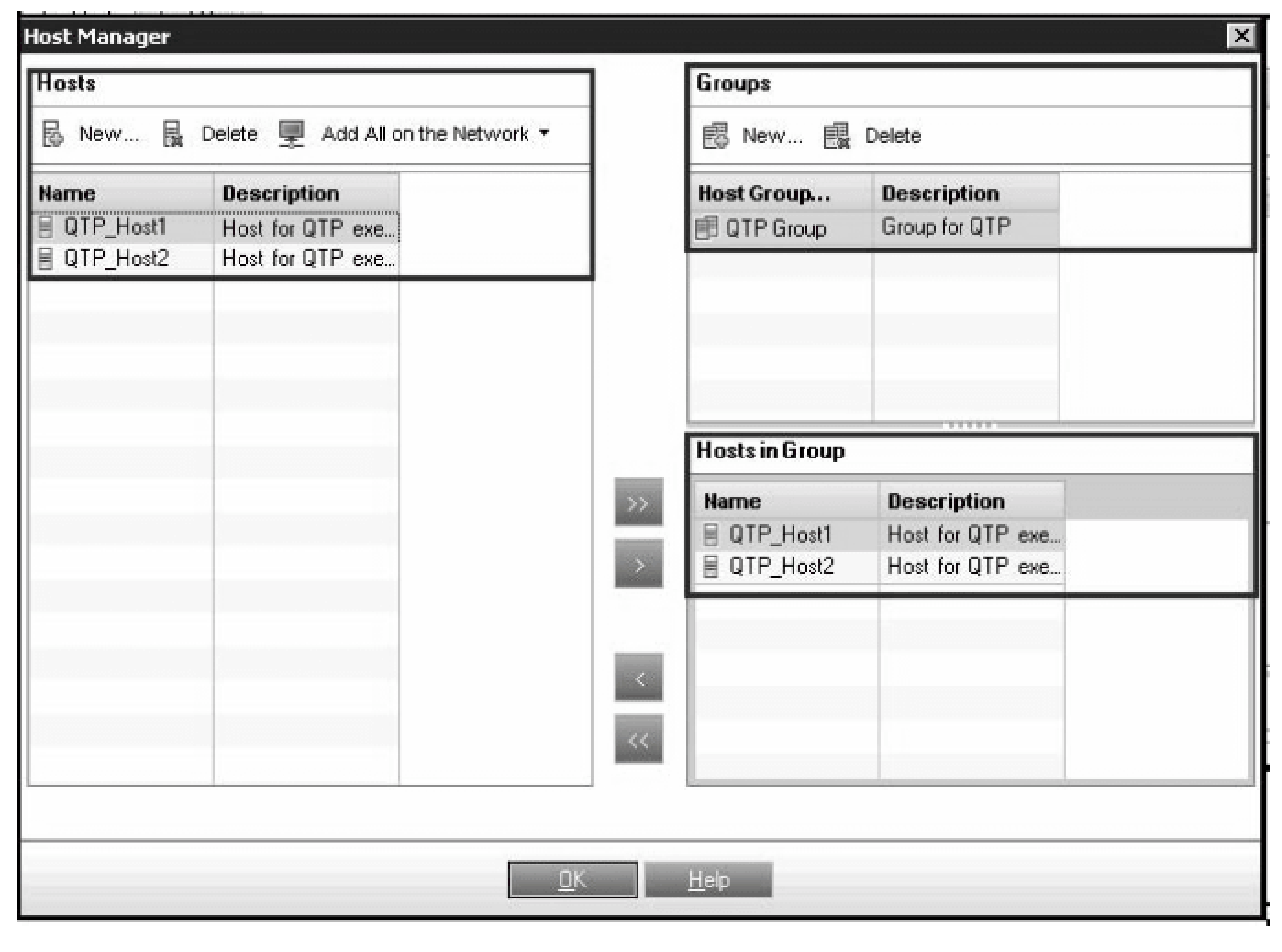


图 12-24　主机管理器

注意：如果为远程测试执行指定主机组，ALM 在组中第一个可行主机上运行，而不是主机组中的任意主机。

1) 选择 Test Set | Host Manager，弹出 Host Manager 对话框。如果 Host 列表中没有显示主机，单击 Add All on the Network 按钮。ALM 扫描网络邻居目录，将找到的主机添加到 HOST 列表。使 Host 列表与网络邻居目录同步，单击 Add All on the Network 箭头，选择 Synchronize Host in the Project With Host on Net。ALM 添加在网络邻居目录里找到的主机，删除在网络邻居目录里未找到的。

2) 将主机添加到 Host 列表，单击 New Host 按钮。弹出 New Host 对话框。在 Host Name 处，输入 Host Machine 的名字。在 Description 处，输入主机描述，单击 OK。

3) 在 Hosts 列表中删除主机，选择主机，单击 Delete 按钮。单击 Yes 确认。

4) 创建主机组，单击 New Host Group 按钮。弹出 New Hosp Group 对话框。在 Group Name 处，输入主机组名字。在 Description 处，输入主机组描述。单击 OK。

5) 向主机组中添加主机，在 Group 列表中选择一个组。在 Hosts 列表中选择主机，单击 Add Host to Host Group 按钮。

6) 从主机组中移除主机，选择 Group 列中的一个组。选择 Hosts in Group 列中的主机，单击 Remove Host From Host Group 按钮。

7) 删除主机组，选择 Host Group 列中格的组，单击 Delete Host Group 按钮。单击 Yes 确认。

8) 单击 Close，关闭 Host Manager 对话框。

12.9　查看测试运行结果

通过 Execution Grid 页，查看自动测试的执行结果(如图 12-25 所示)。Execution Grid 页包含 Last Run Report 窗格，显示最近测试运行状态。

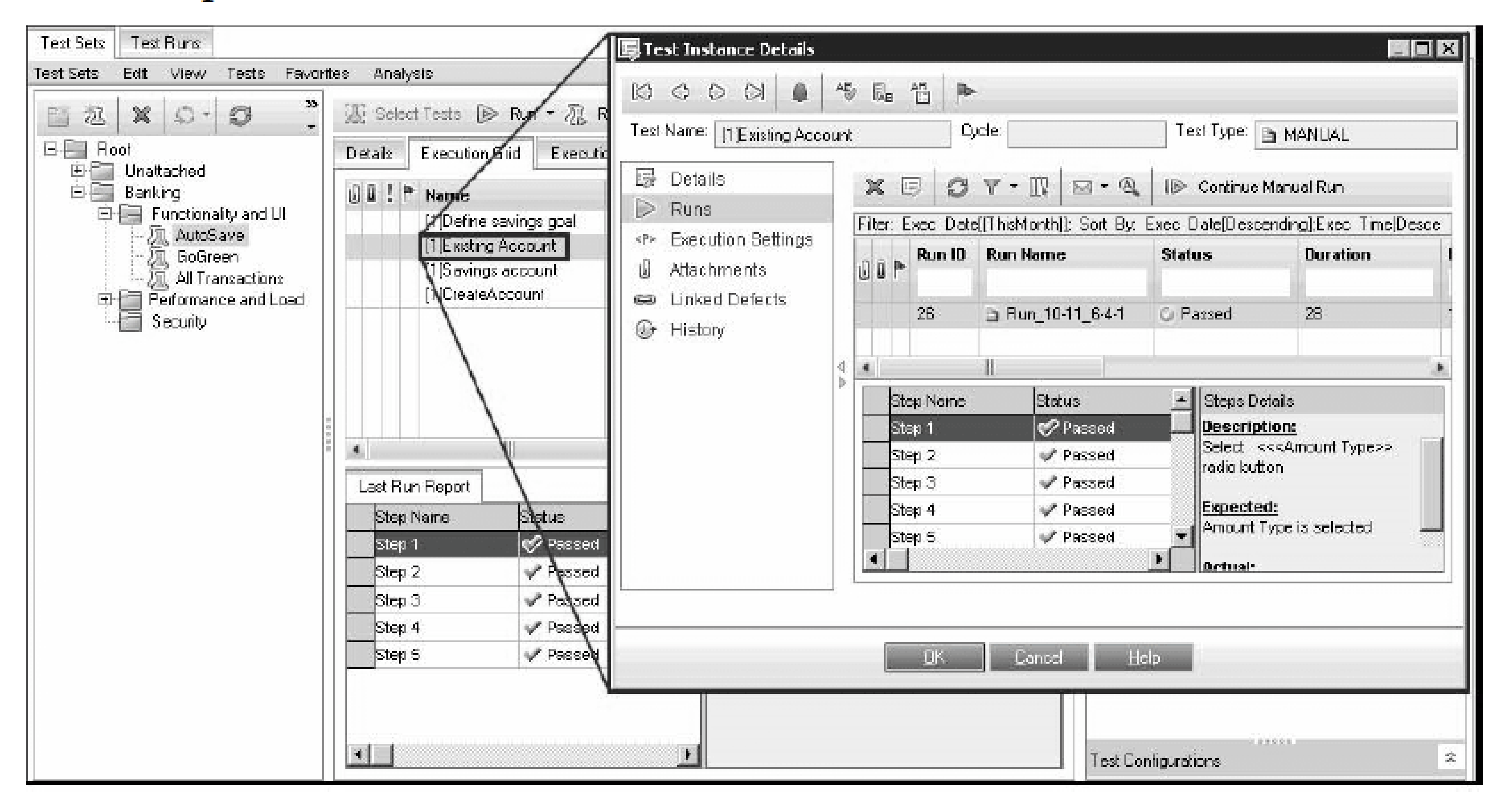


图 12-25　查看测试结果

Last Run Report 显示了所选测试上次运行以来的执行结果。Step Details 显示选中步骤的细节。

对于自动测试，Last Run Report 窗格显示一个额外的按钮 Launch Report，打开所使用的测试工具生成的报表。例如，使用Launch Report 按钮打开 LoadRunner Analysis 中的LR-SCENARIO 测试结果。

在 Execution Grid 页，如图 12-25 所示，显示了测试集最新的运行结果。网格上边的部分显示了测试集中每个测试的结果。下边的部分显示了当前选中测试每一步的结果。Test Instance Details 对话框显示了 Exiting Account 测试执行一次，并且通过。

12.10　检查发布周期过程

显示发布周期过程，ALM 能够监控发布周期的测试过程。偶尔会需要检查发布周期中需要运行的测试实例数量。或者检查实际测试执行率是否与要求的测试执行率一致。使用 Releases 子模块可以分析这些信息。

检查发布周期过程如图 12-26 所示。

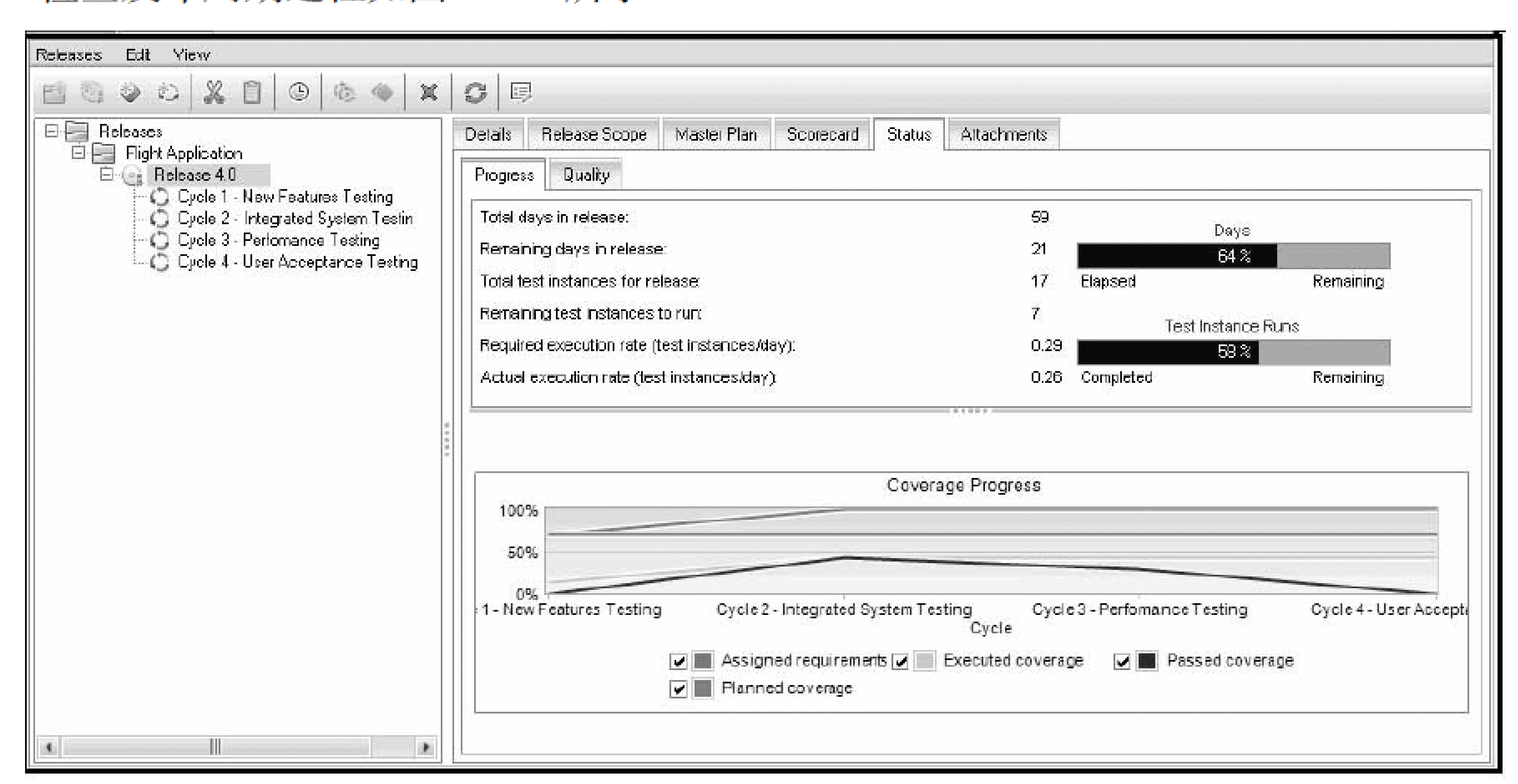


图 12-26　显示发布周期过程

1) 在 ALM 的工具条中单击 Management，然后选择 Releases 标签。

2) 在发布树中，展开 Releases 文件夹，选择一个周期。

3) 在右侧面板，单击 Progress 标签。显示以下字段：

(1) Total days in release：显示选中周期的总天数。

(2) Remaining days in release：显示选中周期剩余的天数。

(3) Total test instances for release：显示与选中周期一致的测试实例总数。

(4) Remaining test instances to run：显示选中周期仍然需要运行的测试实例数目。

(5) Required execution rate(test instances/day)：显示周期剩下天数中每天需要进行的测试

实例数量。

(6) Actual execution rate (test instances/day)：显示选中周期至今每天测试实例数量。

此外，Progress 标签显示周期内已经过去的时间百分比和测试实例完成百分比。Progress 标签提供测试覆盖数据的图标视图。

习题与思考题

判断下列论述是否正确，并描述原因：

1. 一个中断的测试执行必须从开始重新启动。
2. 可以给周期或者发布分配测试集文件夹。
3. 只能给周期分配测试集文件夹，发布不能。
4. 自动测试只能在本地运行的服务器机器上执行。
5. 如果测试没有分配给测试员，ALM 执行测试时会自动将其分配给列表中的第一个测试员。
6. Manual Runner 强制测试按照指定顺序进行。
7. Automated Runner 只运行自动测试。手动测试必须在 Manual Runner 中执行。

练习：建立并执行测试集

创建测试计划树后，创建测试集来分组 Login、Create Order 和 Open Order 测试。测试需要使用以下执行条件：

1. Login 测试需要最先执行。
2. Create Order 测试只有在 Login 测试执行通过后执行。
3. Open Order 测试 需要在 Create Order 测试执行结束后进行，不管执行结果。

本练习需要完成以下任务：

第 1 部分：建立测试集树

第 2 部分：创建测试集

第 3 部分：修改测试集

第 4 部分：执行测试

(该练习的指导步骤请参见本章正文)

第13章　使用HP Sprinter手动测试

13.1　HP Sprinter 概述

HP Sprinter 从 Test Lab 模块的 ALM 中运行手动测试。在基础模式中，Sprinter 提供了多种工具来辅助手动测试过程。你能：

1. 设置显示选项来查看更多的应用。
2. 在测试运行中编辑参数。
3. 提交缺陷到 ALM，如果你不想中断测试，可以设置默认提示。
4. 创建并注释截图以及录制执行测试的过程。

除了运行基本手工测试之外，Sprinter 的 Power Mode 还能够：

1. 在你的测试应用中，记录并运行宏命令。
2. 在应用中自动键入数据。
3. 使用脚本，该脚本显示了测试中执行的每一个活动。
4. 在多台不同配置的机器上复制你的用户活动信息。

13.2　使用 HP Sprinter

有两种方法可以在 Sprinter 中打开一个测试。一种是在 ALM 的 Test Lab 模块中选择一个测试，并从这里启动 Sprinter(如图 13-1 所示)。另一种，打开桌面上的 Sprinter，连接到 ALM，从 ALM 的 Test Lab 模块中打开测试。

从桌面启动 Sprinter 的步骤如下：

(1) 单击桌面的 Sprinter 快捷方式。HP Sprinter 打开，出现 Open a Test 窗口。关闭该窗口。

(2) 第一次运行应用程序时，可以看到 Getting Started 界面。双击主窗口右下方的 HP ALM Connection 按钮。

(3) 在 Sprinter 的主窗口中，单击“打开”按钮。

(4) 输入地址、用户名和密码，单击“认证”按钮。

(5) 输入 Domain 和 Project，单击 Reconnect on Startup 复选框，因为你可能会经常在同一台 HP ALM 服务器上工作。

(6) 单击 Login 按钮，Sprinter 连接到 HP ALM，出现 Open 对话框。

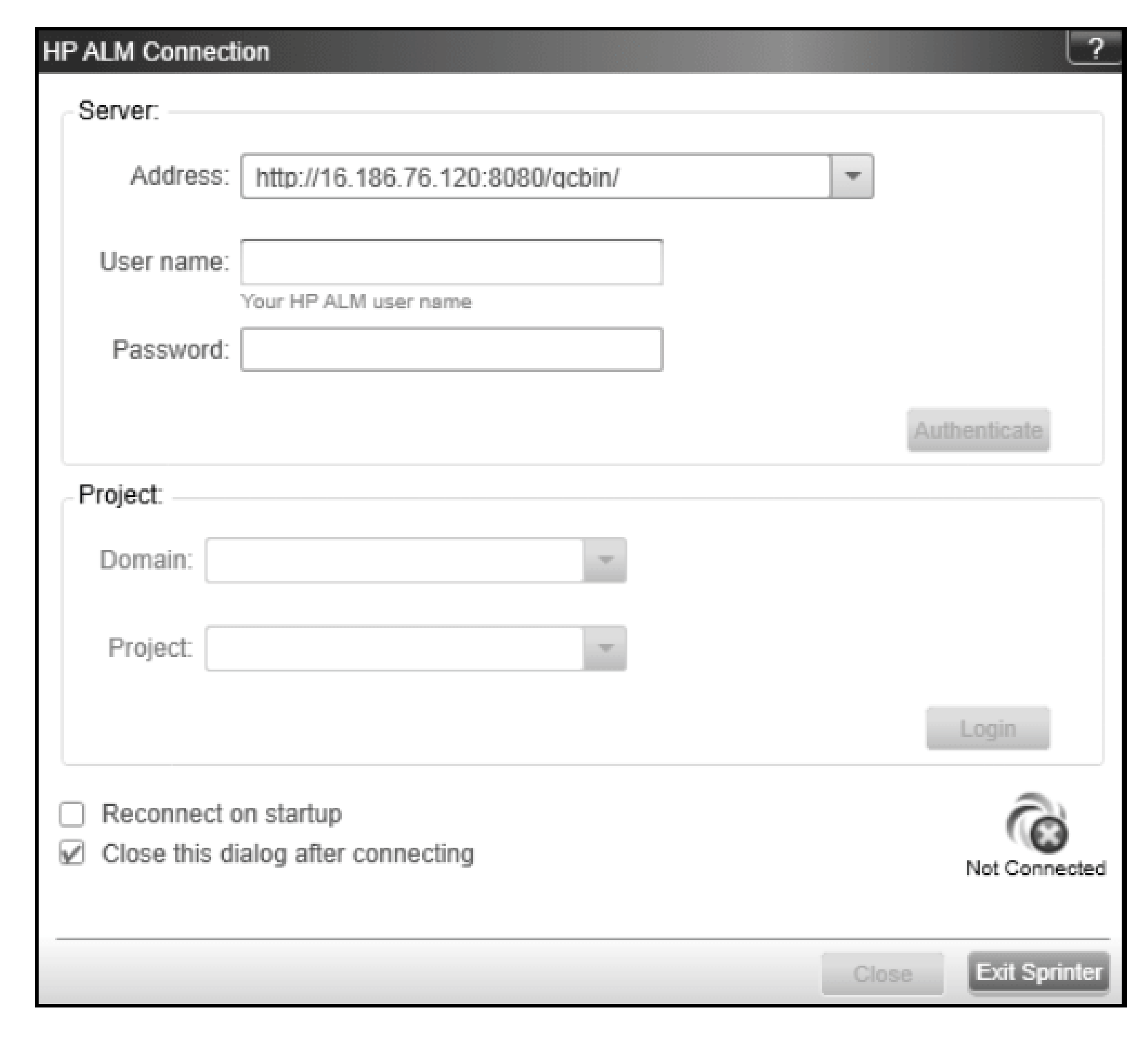


图 13-1　启动 Sprinter

13.2.1　打开测试集

打开一个测试(如图 13-2 所示)的步骤如下：

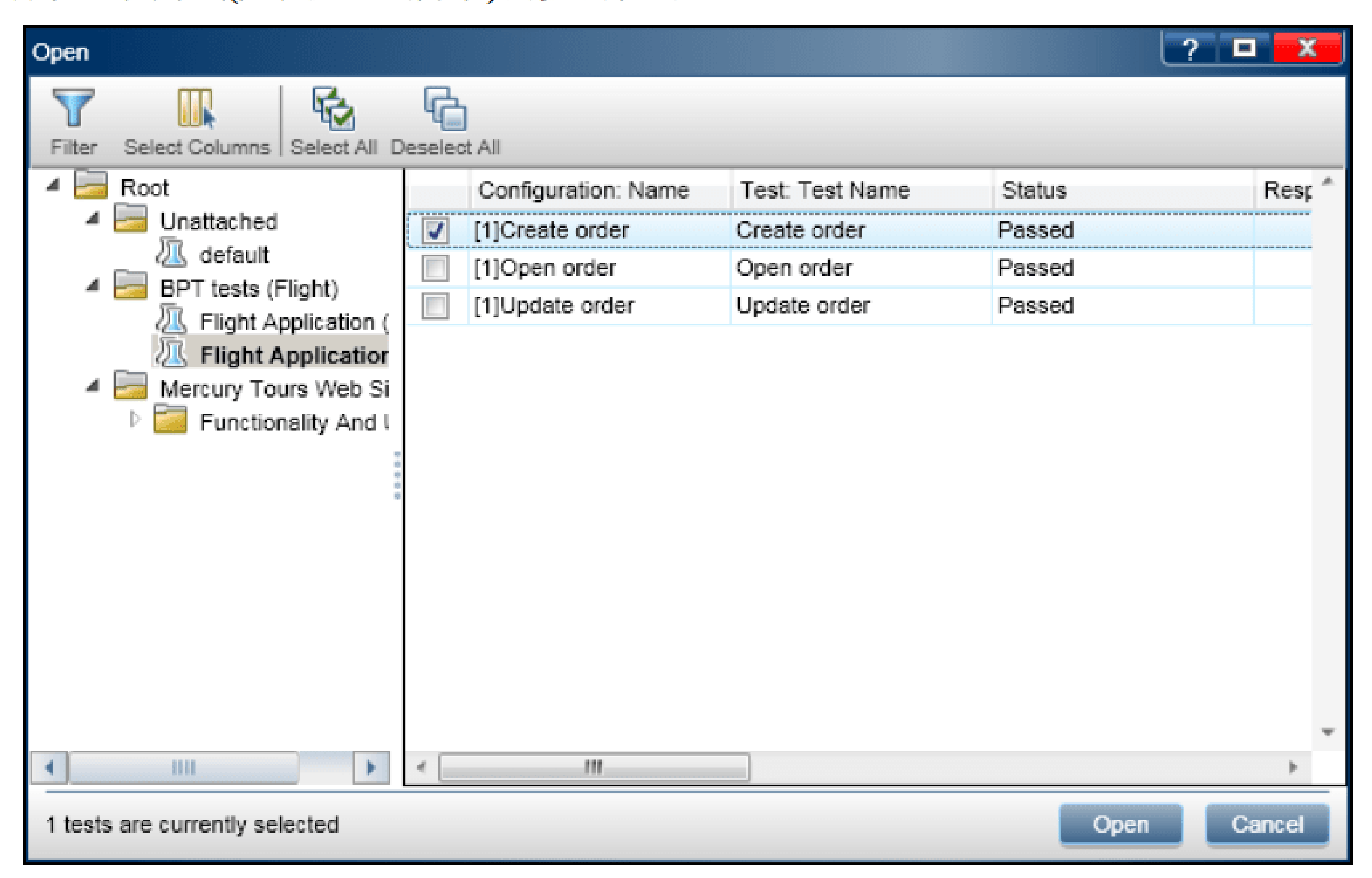


图 13-2　打开测试

(1) 在 Open 对话框中展开测试计划树，直到找到想要执行的测试集。

(2) 选择测试集，然后单击合适的复选框来选择需要运行的测试，或者单击全选按钮来选择所有测试。

(3) 单击打开按钮，出现 Sprinter 窗口。

13.2.2　准备执行测试

Sprinter 窗口包含下列元素(如图 13-3 所示)：

1. 右边窗格显示了 Definitions 组，包括三个节点：General Settings、Steps 和 Parameters 。

2. General Settings 节点显示：测试名称、测试集名称、配置名称、测试所有者、测试描述、运行项名称、测试者名字、运行项状态、日期、运行起止时间、任何附属信息。

3. Steps 节点用于确认测试步骤。

4. Parameters 节点用来确认测试中的参数。

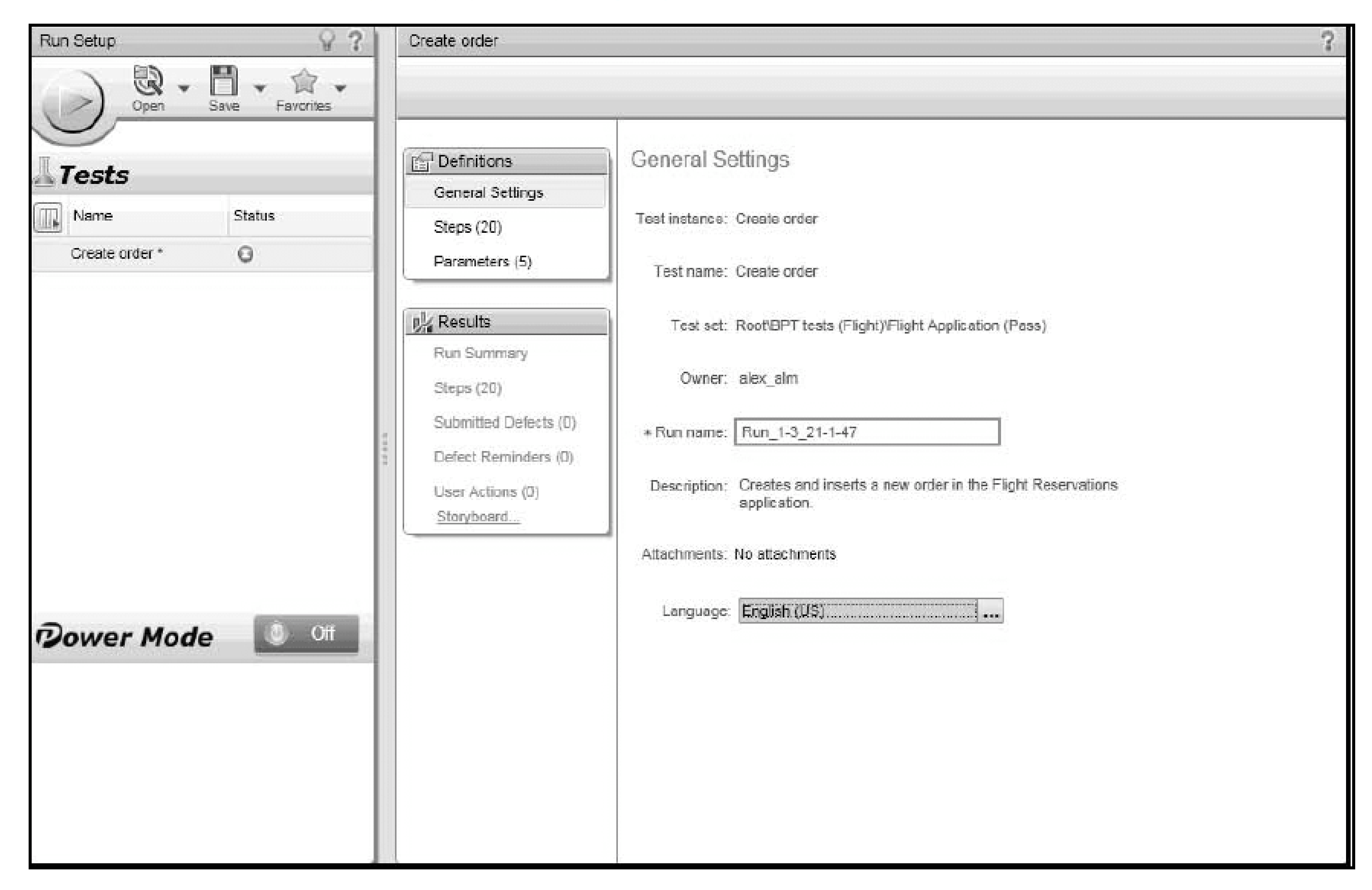


图 13-3　Sprinter 窗口

HP Sprinter 的运行界面如图 13-4 所示。

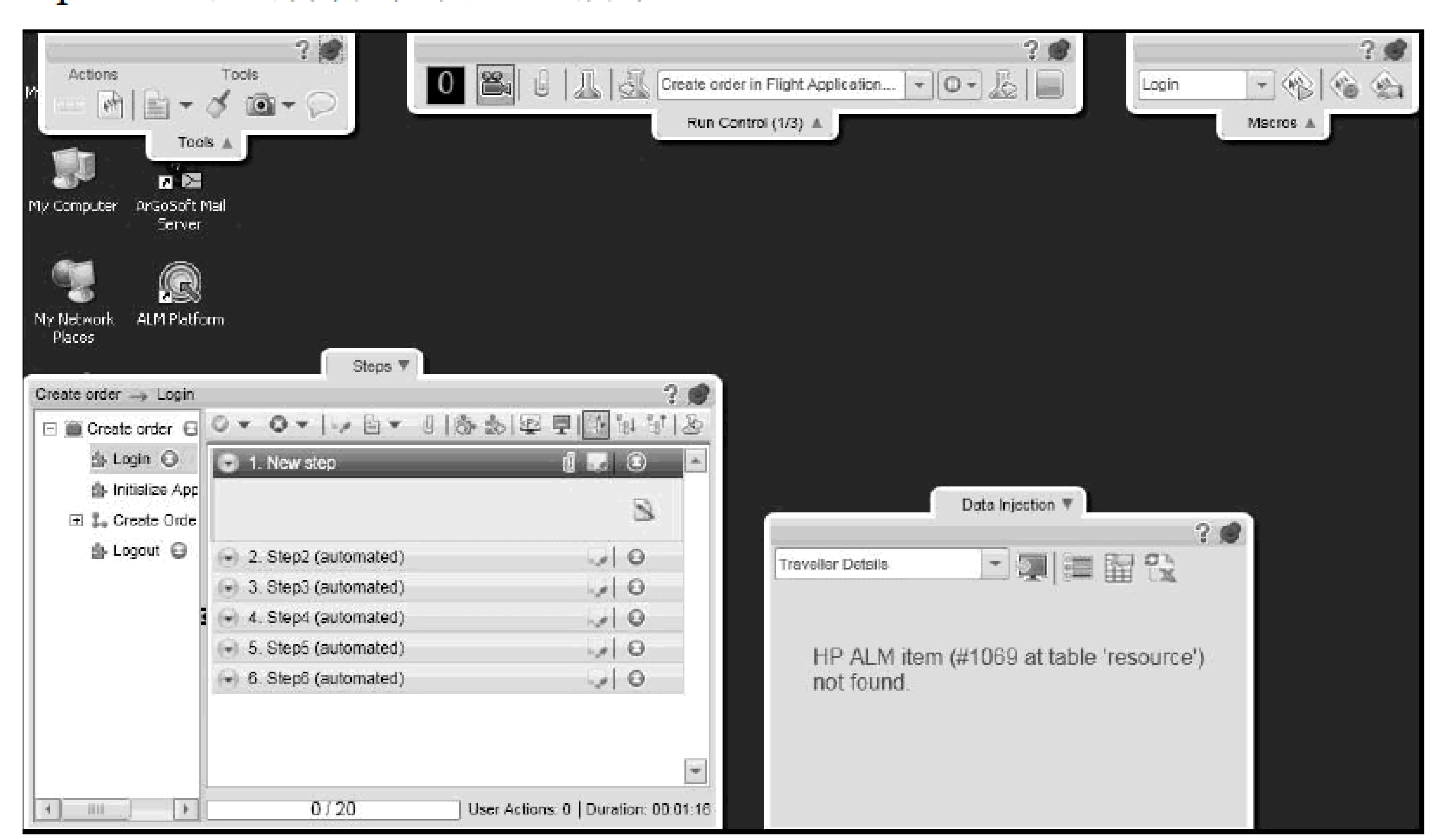


图 13-4　侧边栏

13.2.3　开始基础测试

如图 13-5 所示开始一个基础测试的步骤如下。



图 13-5　开始一个基础测试

(1) 打开测试下的应用程序。

(2) 单击 Run Setup 区域的运行按钮。Sprinter 主窗口最小化，几个侧边栏出现在 Windows 桌面的边缘。

(3) 检验是否有如下侧边栏：1) Tools 侧边栏在 Windows 桌面的上边缘；2) Run Control 侧边栏也在上边缘；3) Steps 侧边栏在下边缘。

按照 Steps 窗口中的说明执行用户交互，评价预期结果，能否通过该步骤，必要时对实际结果进行注释。

执行一个测试的步骤如下：

(1) 自动打开 The Steps 侧边栏。如果没有，单击 Steps 选项卡。

(2) 测试的第一步按照说明进行。在这个例子中输入 Agent Name。

(3) 评价应用程序是否如预期一样进行，然后从工具条里选择是否通过(Pass 或 Fail)。

(4) 继续进行步骤，直到完成。

13.2.4　停止测试

如图 13-6 所示停止运行项并查看测试结果的步骤如下：

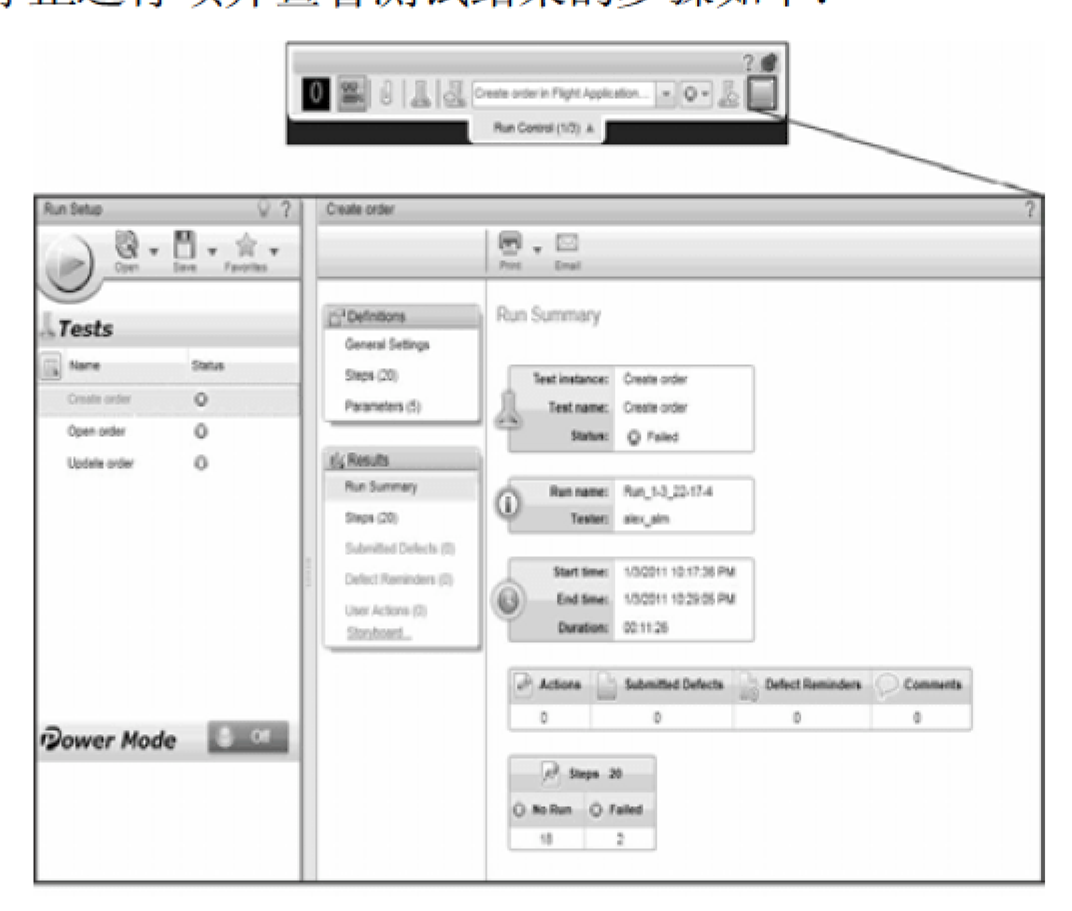


图 13-6　停止测试

(1) 展开 Run Control 侧边栏，单击 End Run 按钮来停止执行。Sprinter 主窗口还原。

(2) 在 Sprinter 主窗口的 Run Summary 区域检查测试结果。

13.2.5　设置参数

如图 13-7 所示设置参数的步骤如下：

(1) 选择这个例子中的第二个测试 Search by Order No.。

(2) 单击 Definition 组中的 Parameters 链接。

(3) 输入参数 Actual Value 的值。

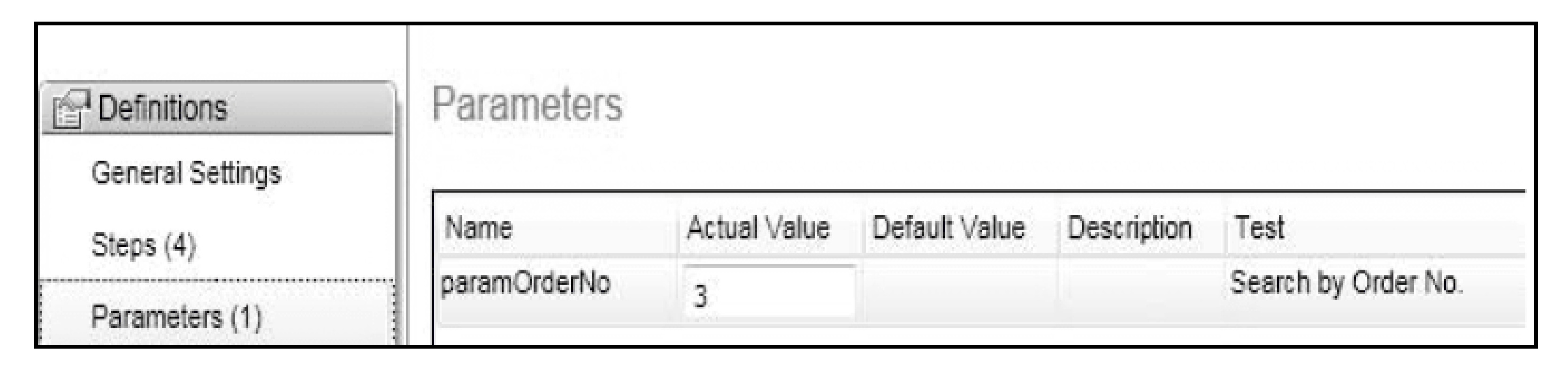


图 13-7　设置参数

13.2.6　使用副标题

对于下一个测试，我们将用 Subtitle Feature 来运行。这使得我们可以用最小化的测试接口来运行该测试，并减少对应用程序窗口的干扰。

如图 13-8 所示显示 Subtitles 的步骤如下：

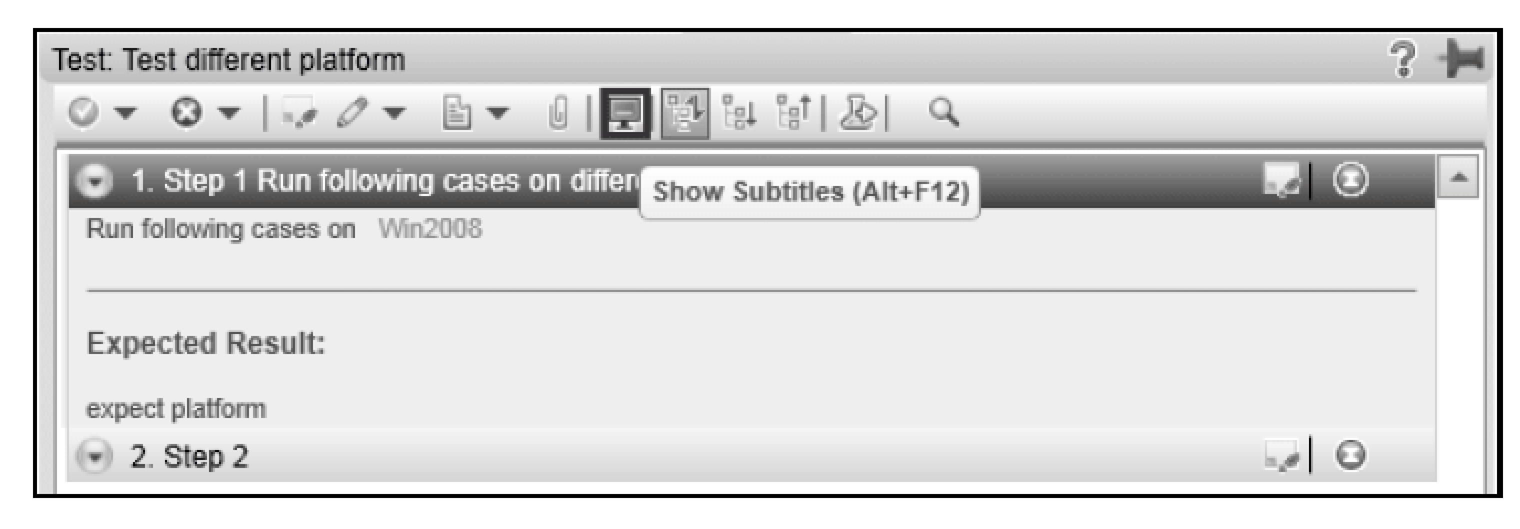


图 13-8　副标题

(1) 单击 Run 按钮以激活测试集中的下一个测试。

(2) 单击 Steps 侧边栏的 Show Subtitles 按钮。Subtitles 模式以字幕形式在屏幕上显示每一步骤的描述，而不是以侧边栏的形式。

(3) 单击 Subtitle，然后就会出现一个 Subtitle 菜单栏。你可以为每一步选择 Pass 或者 Fail。

(4) 单击 Subtitles Settings 按钮，出现 Subtitles Settings 对话框。用该对话框来更改 Subtitles 的设置以简洁可见。

(5) 用 Pass 或 Fail 按钮来标记每个步骤的状态。同时，你也能用 The Drop Down With The Status Button(状态按钮的下拉菜单)状态的可能值有 Passed、Failed、Blocked、Not Completed、No Run、N/A。

13.2.7　记录实际结果

如图 13-9 所示记录实际结果的步骤如下：

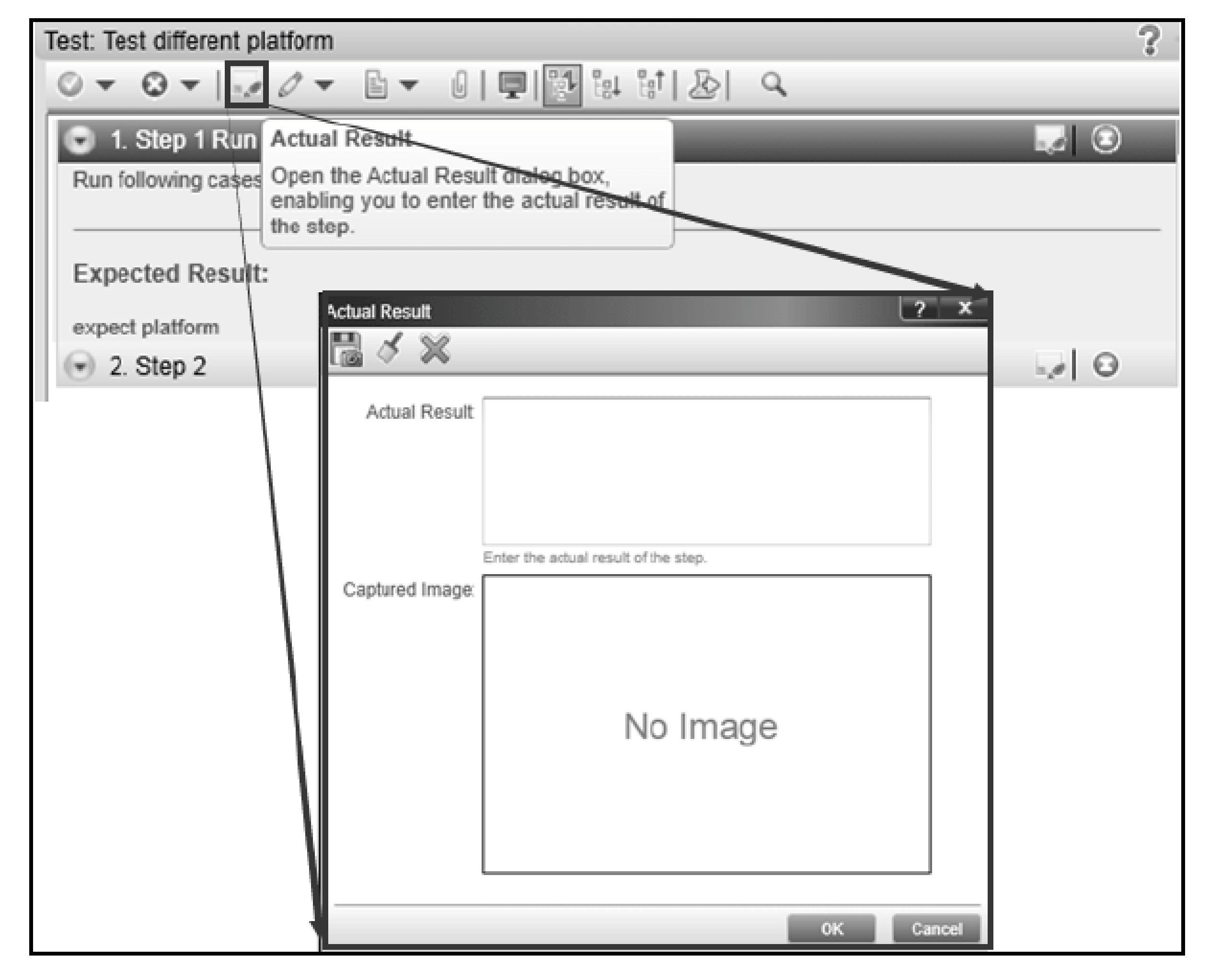


图 13-9　记录实际结果

(1) 当实际结果和预期结果不一致时，单击 Actual Result 按钮，出现 Actual Result 对话框，输入文本对实际结果进行描述。

(2) 如果要在实际结果中插入图片，请单击 Save Screen Capture as Actual Result 按钮，然后单击 Actual Result 对话框中的 OK 按钮。如图 13-10 所示。

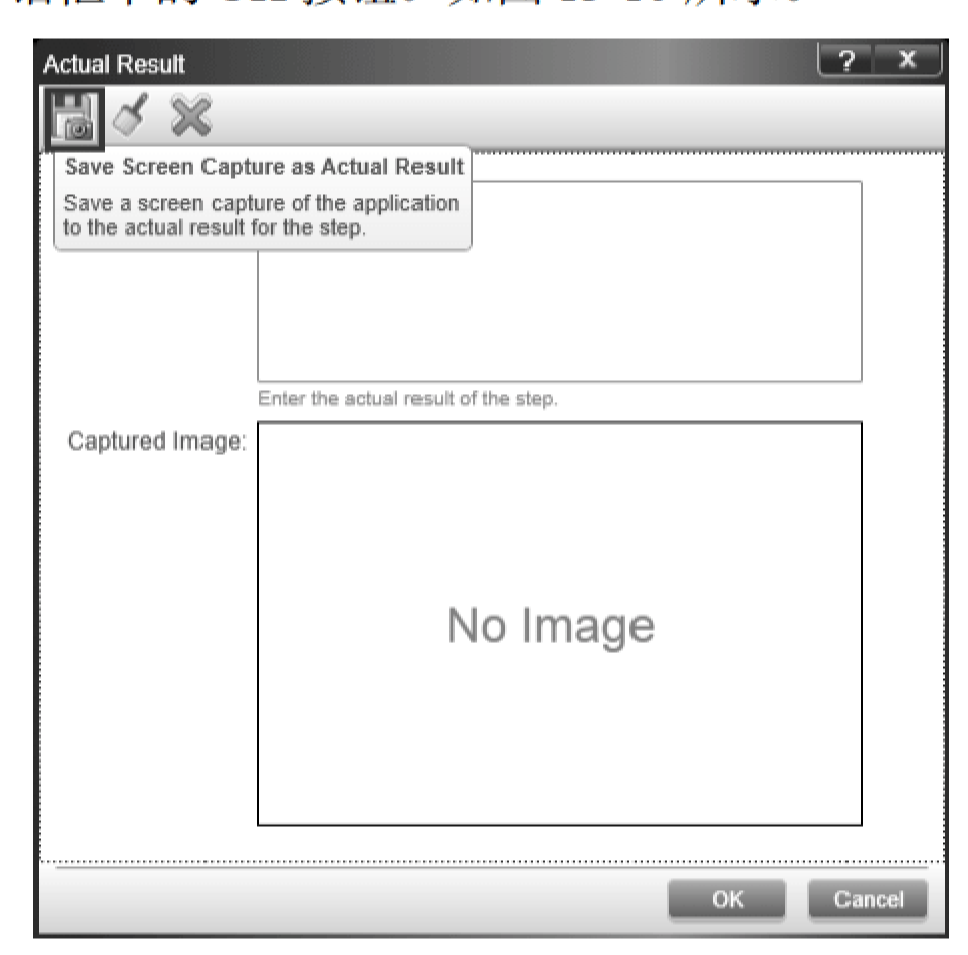


图 13-10　插入图片

添加一个新的测试步骤或者编辑一个已有的测试步骤：

(1) 从Steps工具栏中单击Edit Steps 下拉菜单要添加一个新的测试步骤，选择Insert Before 或者 Insert After 选项，弹出 Edit Steps 对话框，输入 Name、Description 和 Expected Results，单击 OK，关闭 Edit Steps 对话框。新的测试步骤就出现在 Steps 工具栏的列表中。

(2) 选择 Edit Steps 选项来编辑一个已经存在的测试步骤。在 Edit Steps 对话框中修改 Name、Description 和 Expected Results 的值之后，单击 OK，关闭 Edit Steps 对话框。

13.3　提交缺陷

如图 13-11 所示输入 Smart Defect 的步骤如下：

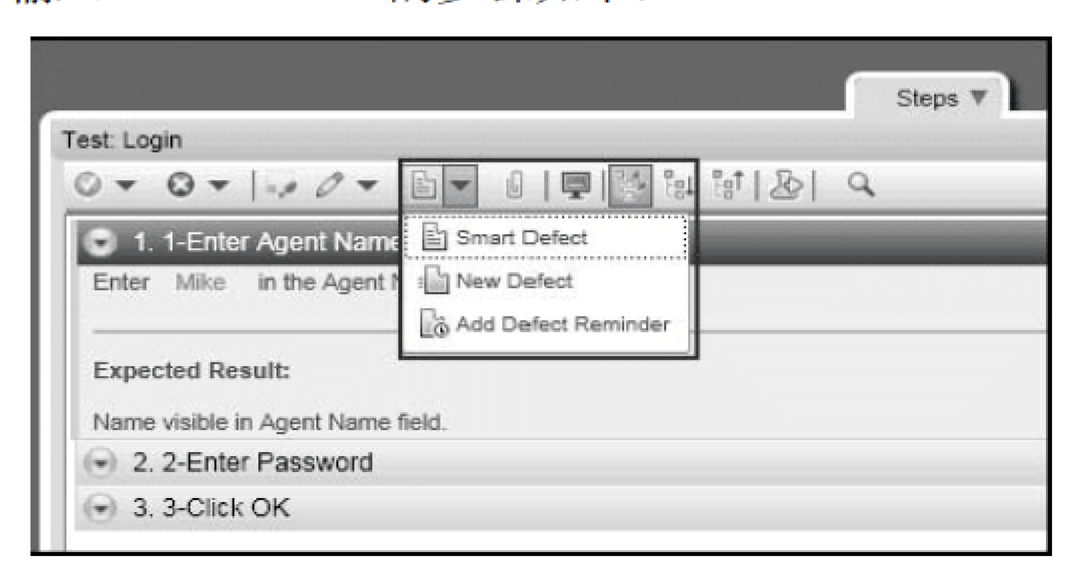


图 13-11　提交缺陷页面

(1) 单击 Smart Defect 按钮，打开下拉菜单，选择 Smart Defect 选项。该选项包括用户活动列表和一段导致缺陷的运行录像。

(2) 当程序在运行过程中，如果发现了一个缺陷但又不想中断运行时，可以使用 Add Defect Reminder，这个提示包含在测试结果中，并且可以在测试结束后查看。稍后你可以提交测试结果中的这个缺陷。测试过程中可获得的信息同样也能从测试结果中获得。所以你能获得测试过程中关于缺陷的有注释的屏幕捕获、录像、步骤/活动信息。

添加缺陷提示的步骤如下：

(1) 单击 Smart Defect 按钮，打开下拉菜单，选择 Add Defect Reminder 选项。

(2) 在 Defect Reminder 对话框中，输入对于缺陷的描述，然后单击 OK 按钮。

13.4　编辑步骤

如图 13-12 所示添加新的测试步骤或编辑已有的测试步骤：

(1) 选择一个测试步骤；

(2) 从 Steps 侧边栏中单击 Edit Steps 菜单，打开下拉列表；

(3) 如果要加入新步骤，首先选择 Insert Before 或 Insert After 选项，然后选择 Edit Steps 选项，出现 Edit Steps 对话框。在对话框中输入名称、描述和预期结果，然后单击 OK 按钮，关闭 Edit Steps 对话框。新步骤被添加到 Steps 侧边栏的步骤列表中；

(4) 如果要编辑已有步骤，选择 Edit Steps 选项，出现 Edit Steps 对话框。编辑名称、描述和预期结果，然后单击 OK 按钮，关闭 Edit Steps 对话框。

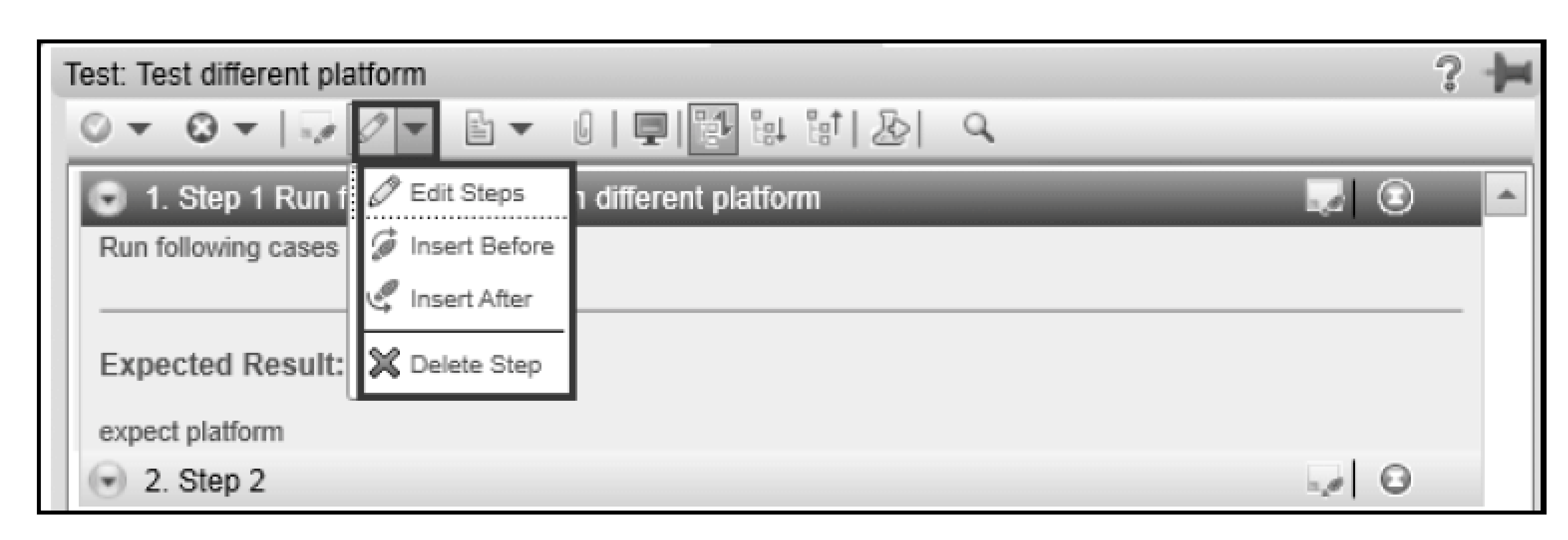


图 13-12　编辑测试图例

13.5　注释屏幕抓图

注释通常用来突出强调一个步骤、运行项、实际结果或缺陷的图像中存在的问题。

如图 13-13 所示注释一个屏幕捕获的步骤如下：

(1) 单击 Actual Result 对话框中的 Save Annotation as Actual Result 按钮，出现 Annotation Workspace。

(2) 利用注释工具标记屏幕捕获。

(3) 单击 Save To Actual Result 按钮。

(4) 单击 Close 按钮。注释过的图像出现在 Actual Result 对话框的图像区域。

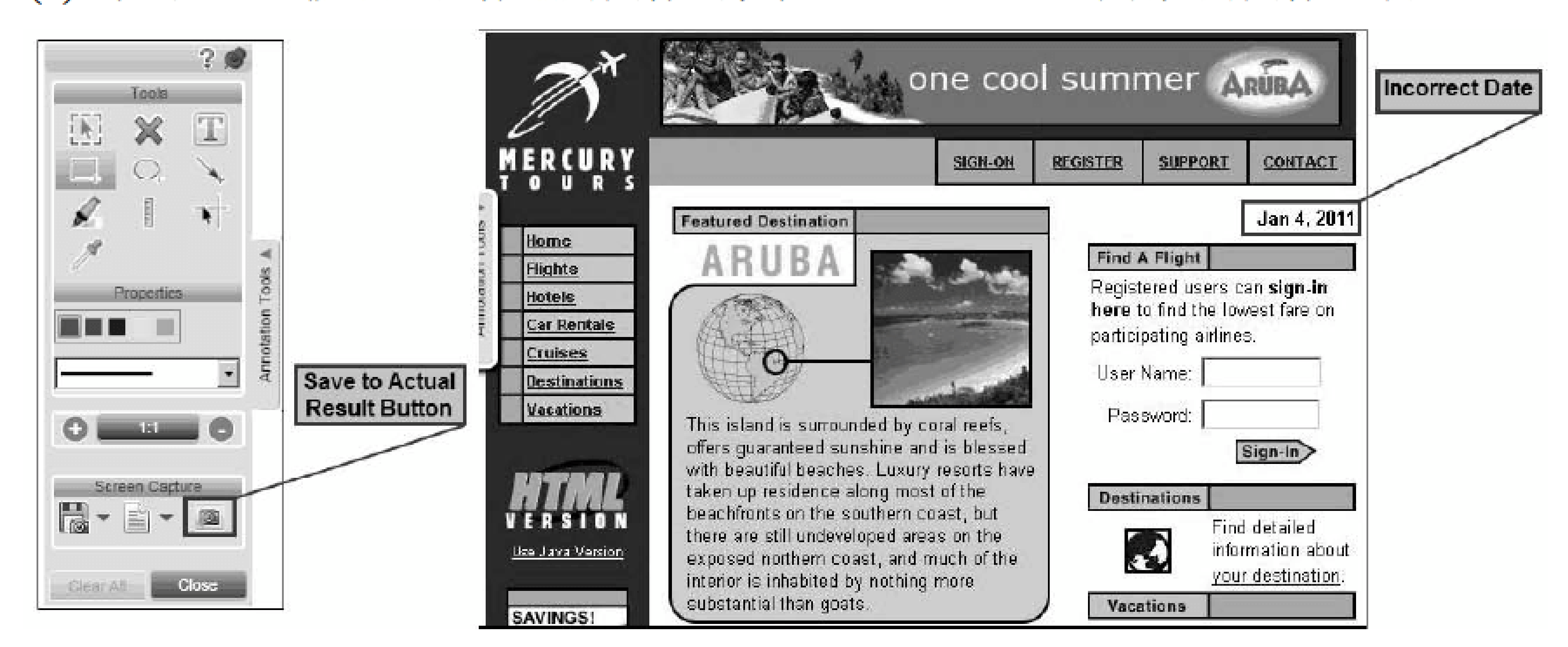


图 13-13　屏幕抓图

13.6　查看运行结果

如图 13-14 所示查看运行结果的步骤如下：

(1) 单击 Run Control 侧边栏的 Stop 按钮来结束运行。侧边栏关闭，Run Summary 窗格出现在主窗口。

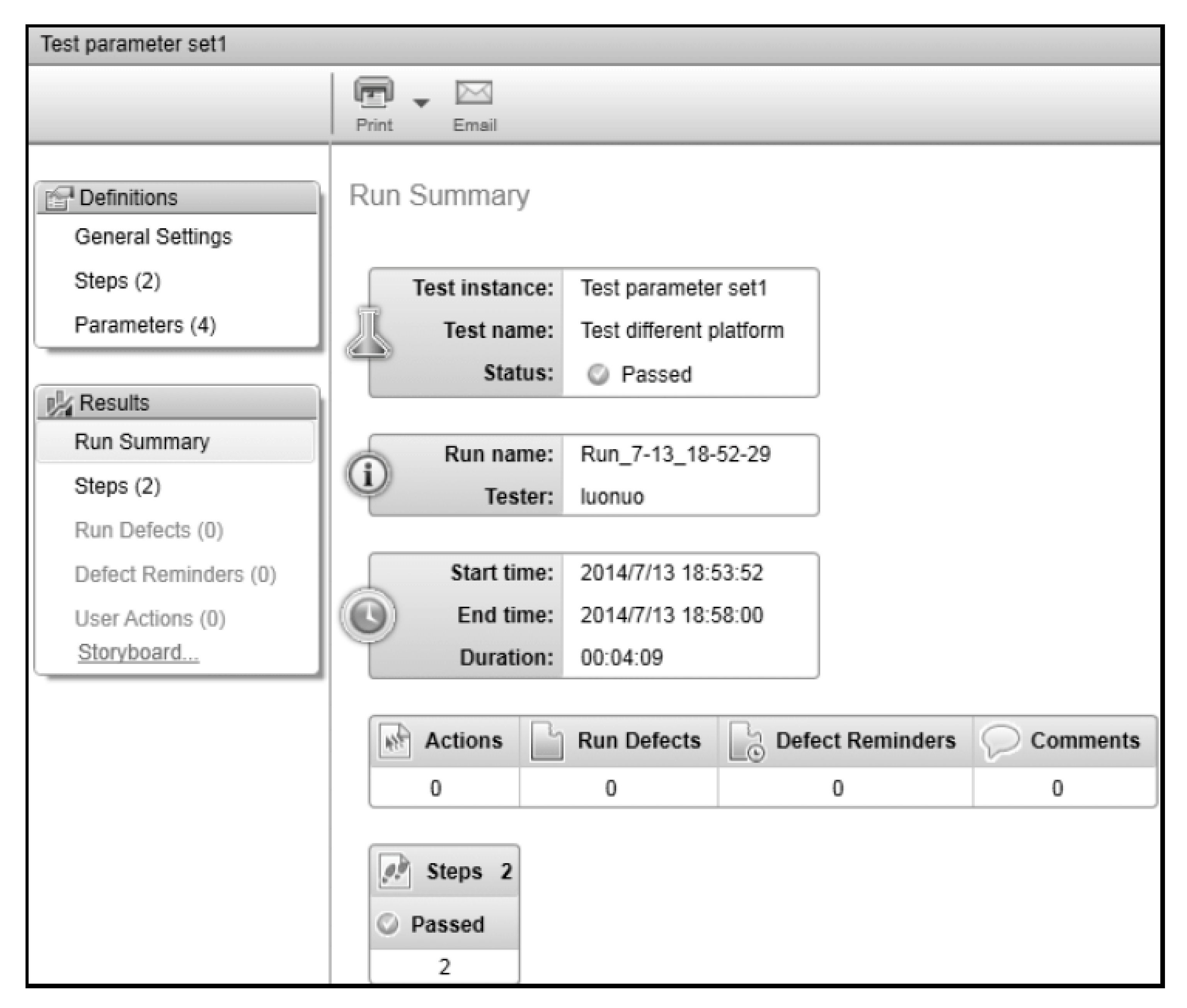


图 13-14　查看运行结果

(2) 摘要包括：测试和运行信息，已执行活动数目(仅在 Power Mode tests 下)，已提交的缺陷数目，已创建的缺陷提示数目，已添加的注解数目(仅在 Power Mode tests 下)，已执行的每一步骤的状态。

(3) 选择 Submitted Defects 节点来查看测试过程中提交的缺陷列表。单击 Defect ID Number 打开对应的 HP ALM Defect Details 对话框。

(4) 选择 Defect Reminders 节点来查看测试过程中创建的缺陷提示列表。你能选择一个提示，单击 Submit Defect 向 HP ALM 提交缺陷。

(5) 选择 User Actions 节点来查看运行过程中执行的用户活动列表。该用户活动列表可以导出为 Excel 数据表。如果在未事先定义的步骤下运行了非正式测试，你可以更改它们，将其作为步骤使用，然后导入 HP ALM 的测试中。

13.7　Power Mode 模式工作

如图 13-15 和图 13-16 所示在 Power Mode 下，Sprinter 捕获应用程序中的每一个用户活动，并以描述性语言保存这些用户活动列表。

(1) Sprinter 获悉的用户活动能被包括在缺陷当中，并且可以在测试结果的 Storyboard 中查看。

(2) Sprinter 记录(每次)应用程序测试过程中的每一时刻的动力模式设置。

(3) Sprinter 提供宏命令以快速通过一些初始界面来获得需要经过严格测试的存储区。

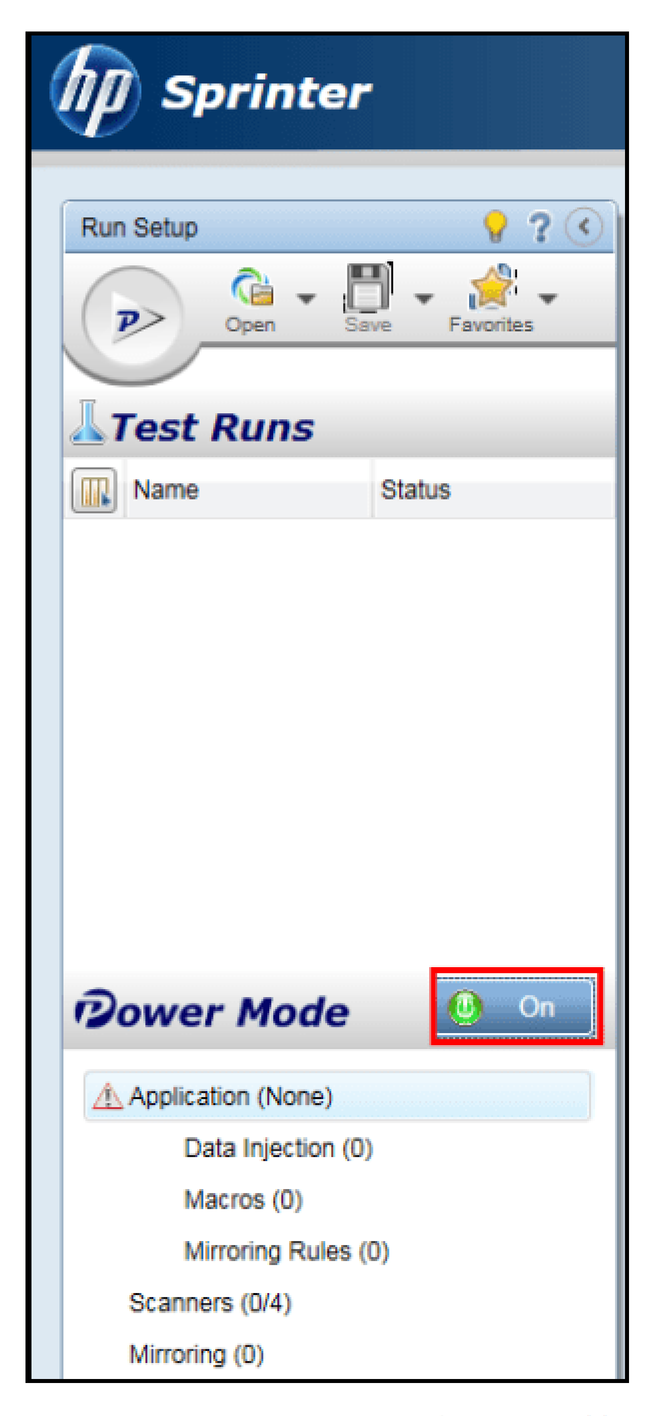


图 13-15　Power Mode 模式工作简介

(4) Sprinter 提供数据注入以填满表格。

1) 单击 Tests 列表下 Power Mode 组中的 On 按钮，出现 The Welcome to Power Mode!对话框。

2) 单击对话框的 On 按钮。

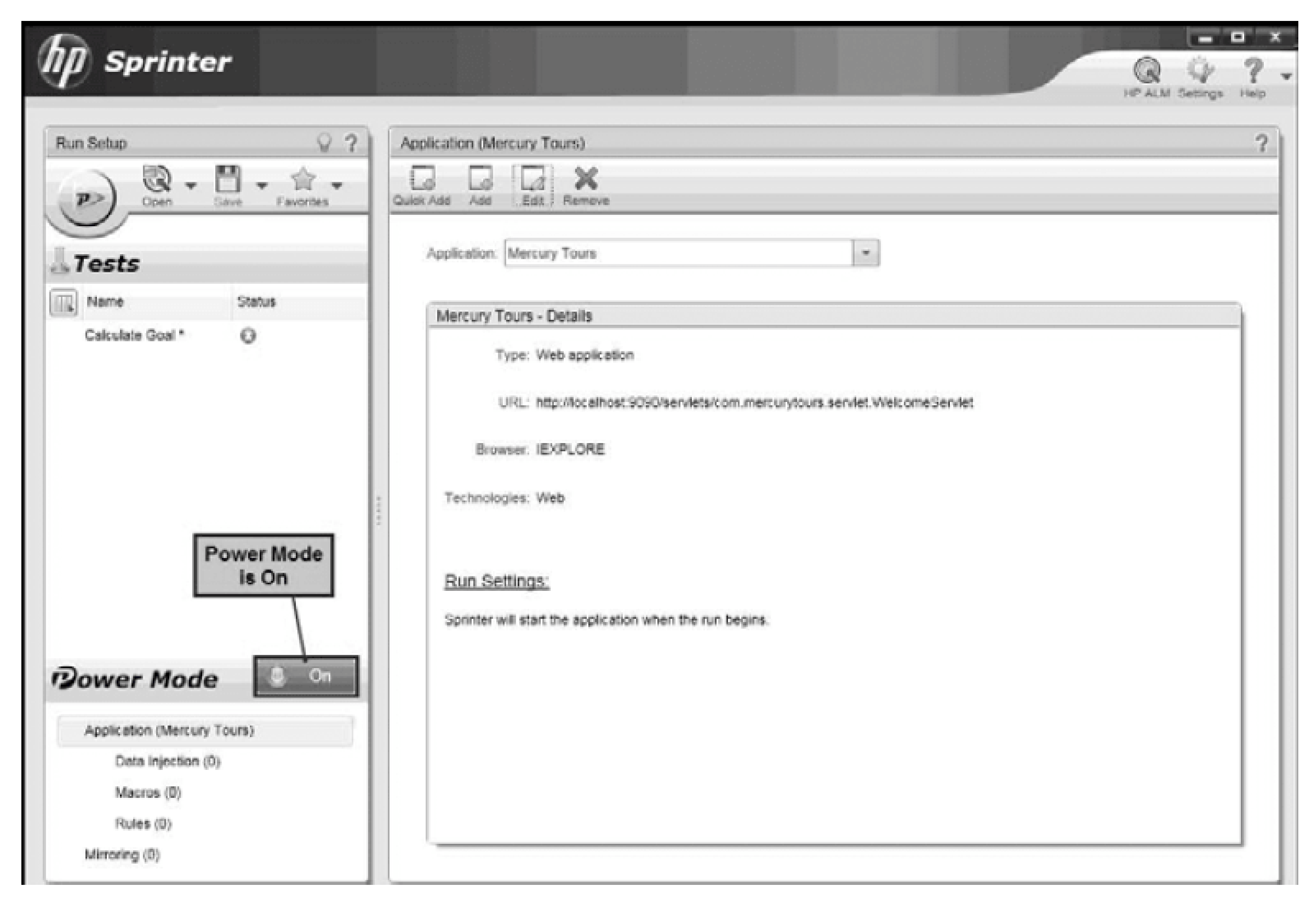


图 13-16　开启 Power Mode

3) 单击 Add 按钮来手动定义一个应用程序。

4) 输入应用名称。

5) 通过单击应用程序的可用类型来改变其类型(单击一个可用的类型来选择应用程序)。

6) 对于一个桌面应用程序，输入 Location 和附加设置。对于网络应用程序，需要提供 URL 以及选择一个浏览器。

7) 选择应用程序中使用的合适技术。

8) 如有必要，选择 Start The Application When The Run Begins 或 Record On Any Open Application。

9) 单击 OK 按钮。

13.8　使用宏命令(Macros)

Macros 允许你记录程序(如图 13-17 所示)，因此你可以快速通过一些初始界面来获得需要经过严格测试的存储区。记录(录制)一个 Macro 的步骤如下：

(1) 单击 Macros 侧边栏。

(2) 单击 Record Macro 按钮。

(3) 执行你想在 Macro 中记录的步骤。

(4) 再次单击 Macro 侧边栏。

(5) 单击 Stop Recording 按钮，出现 Macro Details 对话框。

(6) 输入 Macro 的名字和描述。

(7) 单击 OK 按钮。

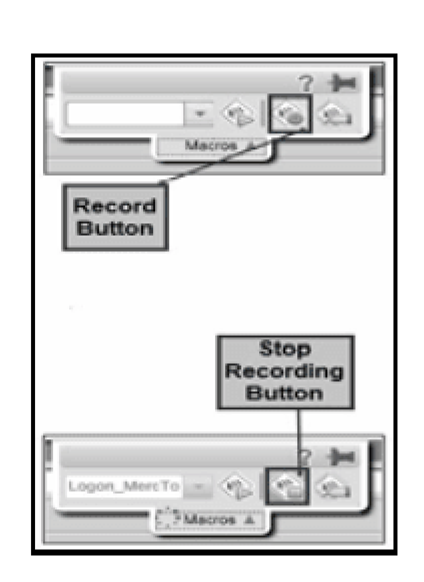


图 13-17　记录 Macros

运行 Macro(如图 13-18 所示)的步骤如下：

(1) 单击 Macros 侧边栏。

(2) 选择 Macros 下拉列表中的一个 Macro。

(3) 单击 Run 按钮。一旦 Macro 运行成功，Macros 标签上会被记上绿色√标记。如果运行不成功，Macros 标签上会被记上红色×标记。

(4) 双击绿色√标记或红色×标记。Macro Status 对话框出现，显示 Macro 运行状态。

(5) 单击 Close 按钮，关闭 Macro Status 对话框。

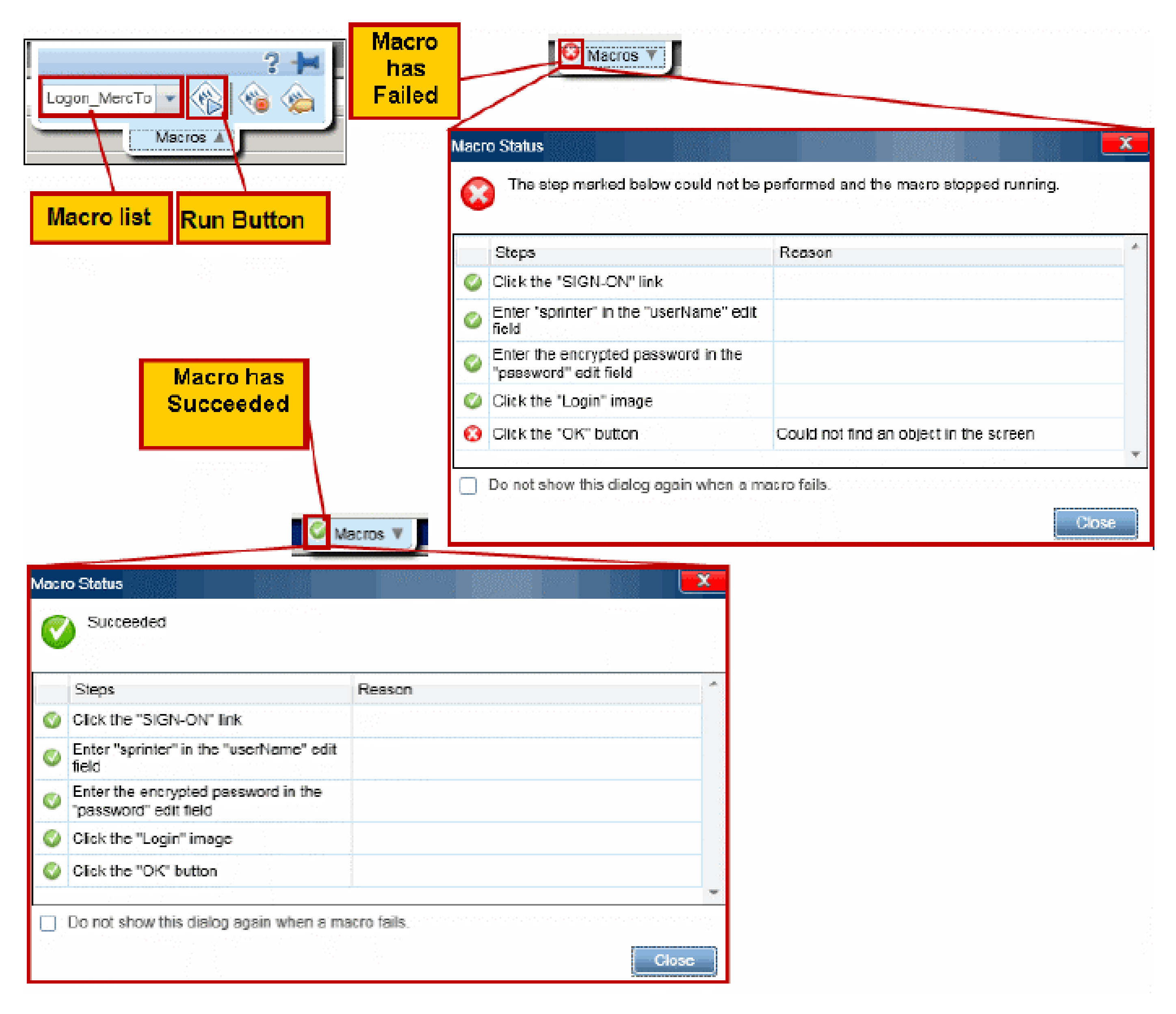


图 13-18　运行 Macros

13.9　使用 Data Injection

如图 13-19 和图 13-20 所示，数据注入能够使被选中的多行数据自动发送到表单中的相关位置。创建用于数据注入的数据集合的步骤如下：

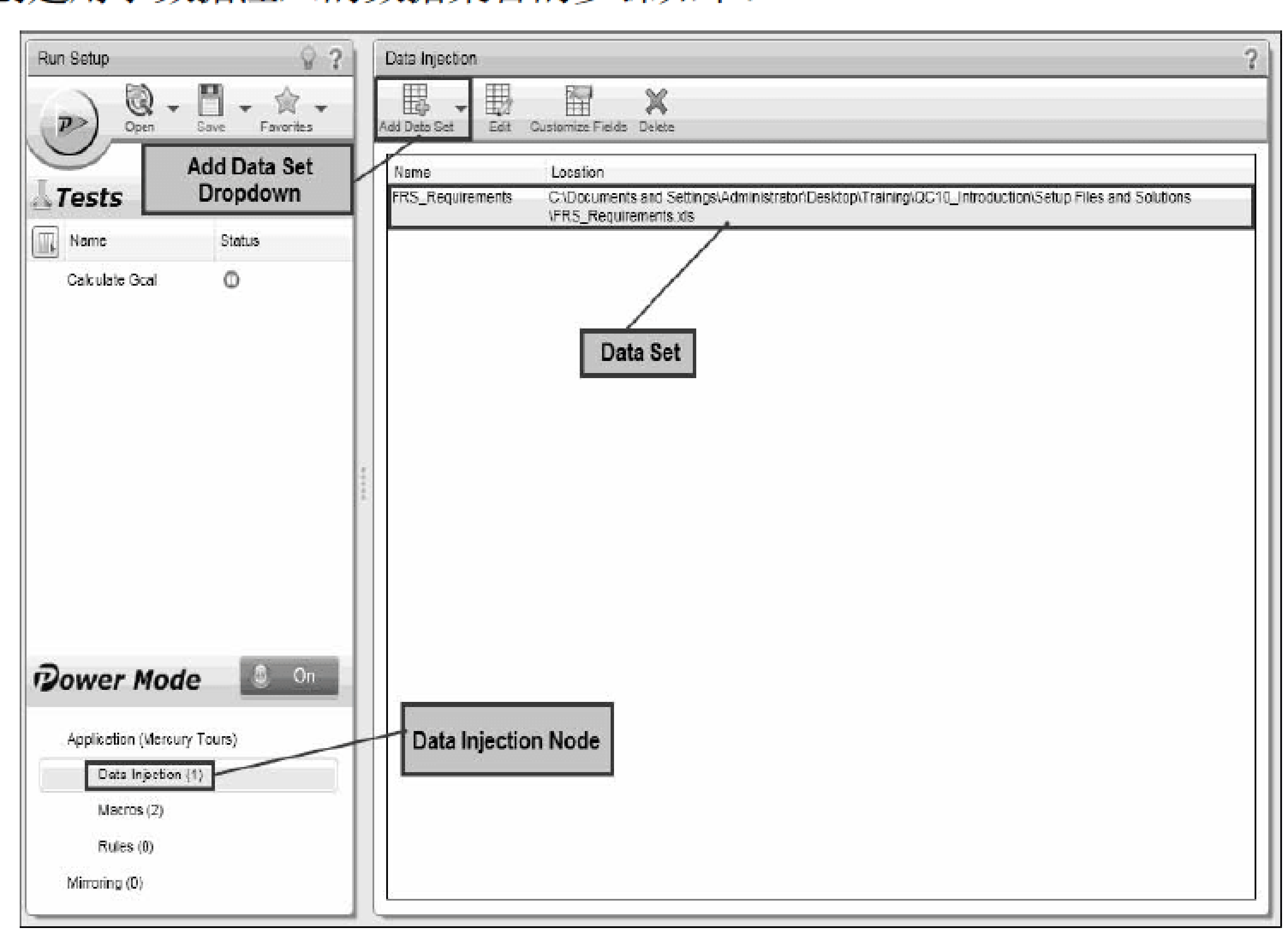


图 13-19　数据注入

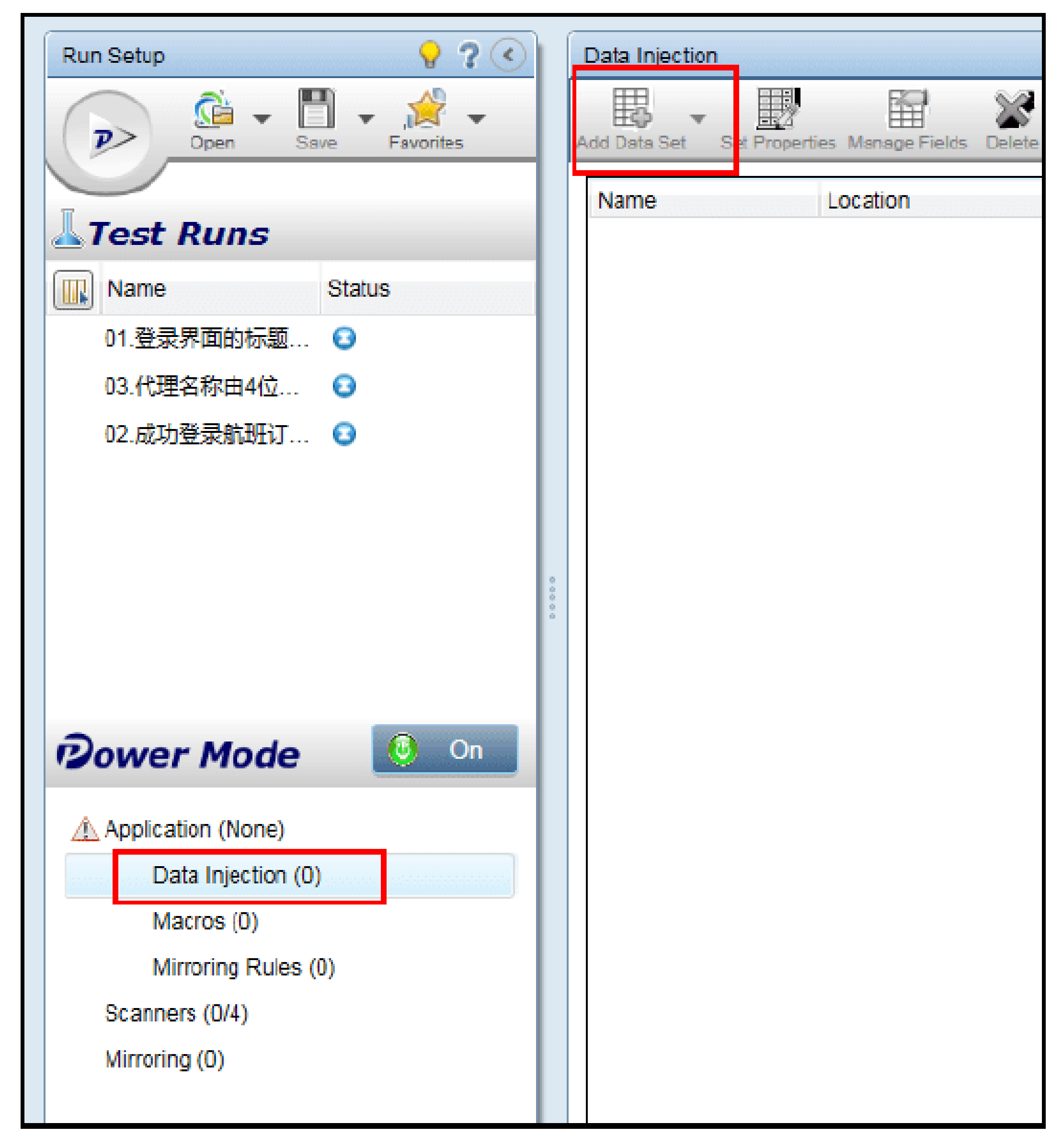


图 13-20　数据注入操作

(1) 单击 Power Mode 组下面的 Data Injection 节点，Data Injection 框架就会出现在右边。

(2) 为了从 HP ALM 中添加一个数据集，需要单击 Add Data Set 下拉菜单并选择 Add from Hp ALM，之后出现一个 Data Set Details 对话框。

(3) 单击 Browse 按钮，会出现一个 Resources 树。展开 Resources 树并在你的系统中查找到 Data Set。

(4) 单击 Open 按钮，则资源的名字以默认名称出现。

(5) 单击 OK 按钮，选中的 Data Set 就会被添加到 Data Injection 框架中去。

在 Power Mode 中使用数据注入的步骤如下：

(1) 单击 Run The Active Test 按钮，则会出现一个 Data Injection 工具栏。

(2) 单击 Data Injection 工具栏。

(3) 选中一组数据并且单击 Inject Data 按钮。

(4) 屏幕将会一直变灰然后显示“Analyzing…”，直到数据注入完成。如果数据注入成功，则在 Data Injection 选项卡中会出现一个绿色的标记√。如果数据注入失败，那么在 Data Injection 选项卡中会出现一个红色×。

(5) 双击绿色的标记√或者红色×，Data Injection Status 对话框出现并显示其状态。

13.10　脚　　本

在 Sprinter 中有一个可以显示测试过程中所执行的每个动作的 Storyboard 模块。对于每一个动作，都可以看到它的快照，也看到任何在运行过程中报告的缺陷、添加的缺陷提醒以

及注释。如果你在多台电脑中执行了测试，那么你将能够看到在不同的电脑上测试运行结果的差异。如图 13-21 所示。

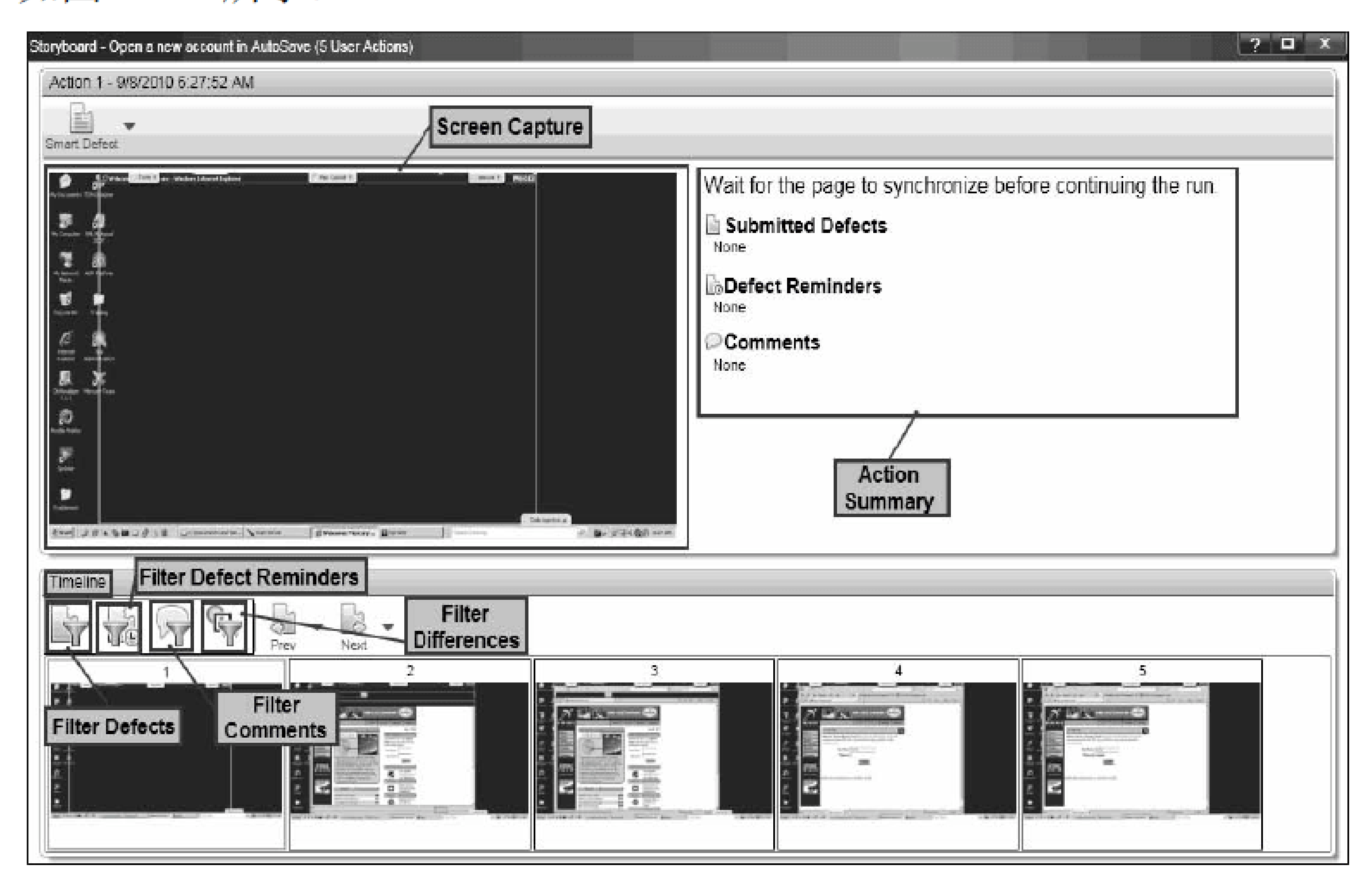


图 13-21　Storyboard

查看 Storyboard 的步骤如下：

(1) 选择 Storyboard 节点，Storyboard 窗口打开。Storyboard 窗口的最上面有一个显示 Timeline 中所执行的用户动作快照面板以及一个 Action Summary 面板，最底部是测试运行的 Timeline。

(2) 在 Action Summary 面板中你可以查看：每一个动作的描述信息，所有已经提交的缺陷，添加的缺陷提醒、缺陷注释。如果你使用镜像运行测试，那么 Action Summary 面板中还会显示不同机器上测试运行的所有差异。

(3) 单击 Action Summary 面板中的链接，打开 HP ALM Defect Details 对话框，基于缺陷提醒创建缺陷或者打开 Differences Viewer。同时，还可以从 Storyboard 中提交一个新的缺陷。

(4) Storyboard 窗口的底部显示的是测试的 Timeline，Timeline 包括在测试过程中的每一个用户动作的快照，可以使用不同按钮来过滤 Timeline 中显示的快照，比如使用 Filter Defects 按钮来使得 Timeline 中只显示你提交过缺陷的动作快照，使用 Filter Defect Reminders 按钮来显示创建了缺陷提醒的动作快照，使用 Filter Comments 按钮来显示添加了注释的动作快照，使用 Filter Differences 按钮来显示那些存在差异的动作快照。

13.11　使用镜像

镜像用来在不同的配置条件下运行相同的测试用例。通过使用镜像，在主机应用程序上执行的每一个用户动作都被复制到定义的另一台电脑上(如图 13-22 所示)。

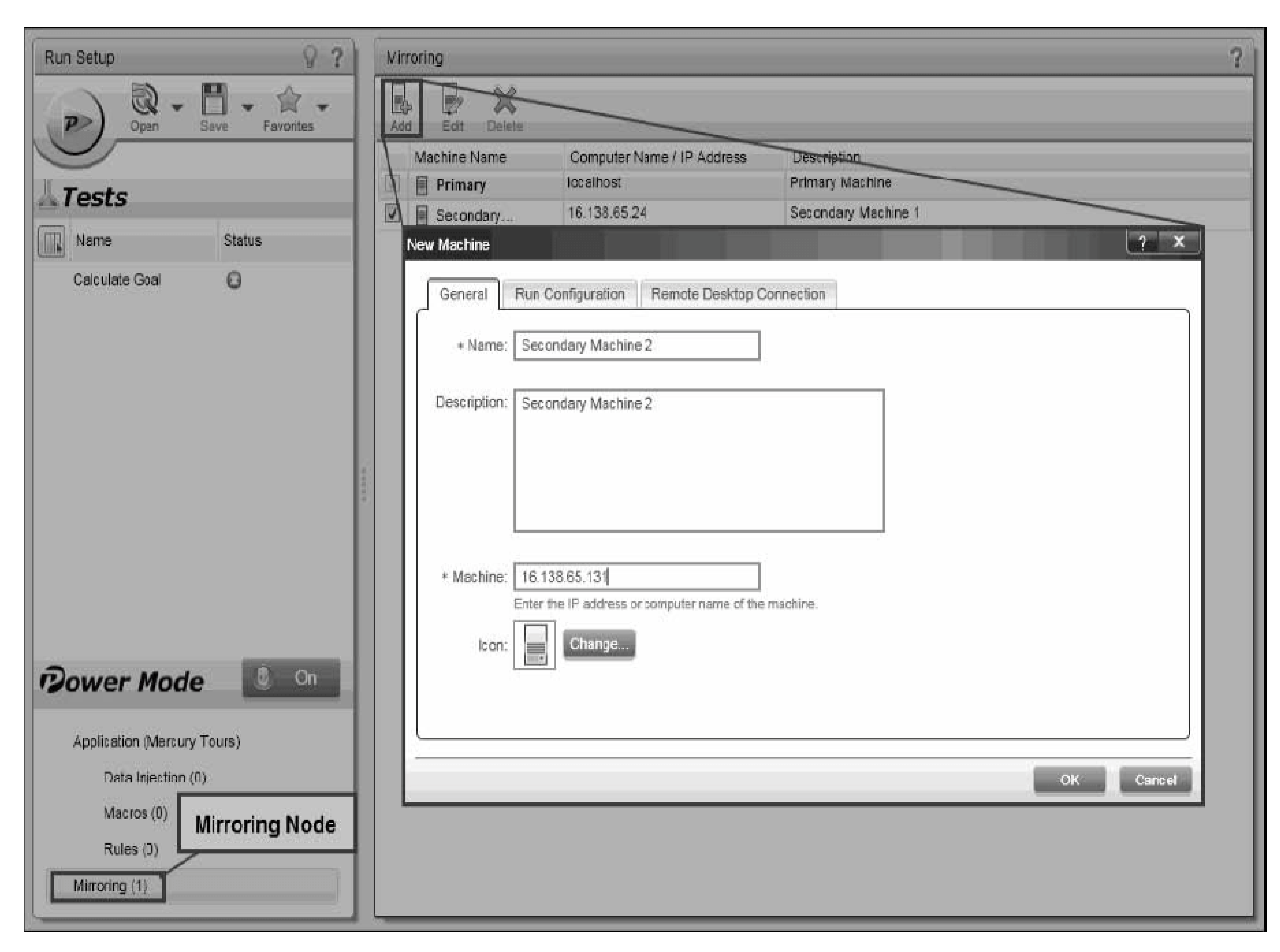


图 13-22 镜像

使用镜像的步骤如下：

(1) 在 Power Mode 中选择 Mirroring 节点。

(2) 单击 Add 按钮来为你的程序添加一台新的机器。

(3) 在 General 中，为作为后备的机器提供了 Name、Description、Machine Name 以及 IP Address 四个属性。

(4) 在 Run Configuration 中，为 Configure How Sprinter Would Launch The Application On The Machine 选择一个选项，然后选择一个浏览器。

(5) 如果你要在测试的时候打开一个远程桌面连接，则进入 Remote Desktop Connection，输入 Domain Name、User Name 和 Password。

(6) 单击 OK 按钮。

习题与思考题

1. 用户的行为是如何存储到 Power Mode 中的？
2. 镜像的作用是什么？
3. 注释使用在什么地方？
4. 使用数据注入的目的是什么？

练习：在 Sprinter 中运行基础测试

你已经在 Test Lab 模块中设置了测试的相关配置，并使用 ALM 手动和自动执行程序测试以及基于被测试应用程序的测试集。

在本次练习中，需要完成以下任务：

第 1 部分：打开一个测试，并准备运行这个测试

第 2 部分：运行测试

第 3 部分：查看运行结果

第 4 部分：使用 Subtitles

第 5 部分：记录缺陷

(该练习的指导步骤请参见本章正文)

第14章　跟踪缺陷

14.1　使用缺陷模式

有效地定位并修复缺陷是开发过程中必不可少的工作。如图 14-1 所示，缺陷跟踪是整个项目周期管理的一部分，也是最后一个环节。使用 HP 的应用程序生命周期管理系统(ALM)的缺陷模块，可以报告程序的设计缺陷，并在整个应用程序管理过程中的阶段都追踪来源于缺陷记录的数据。

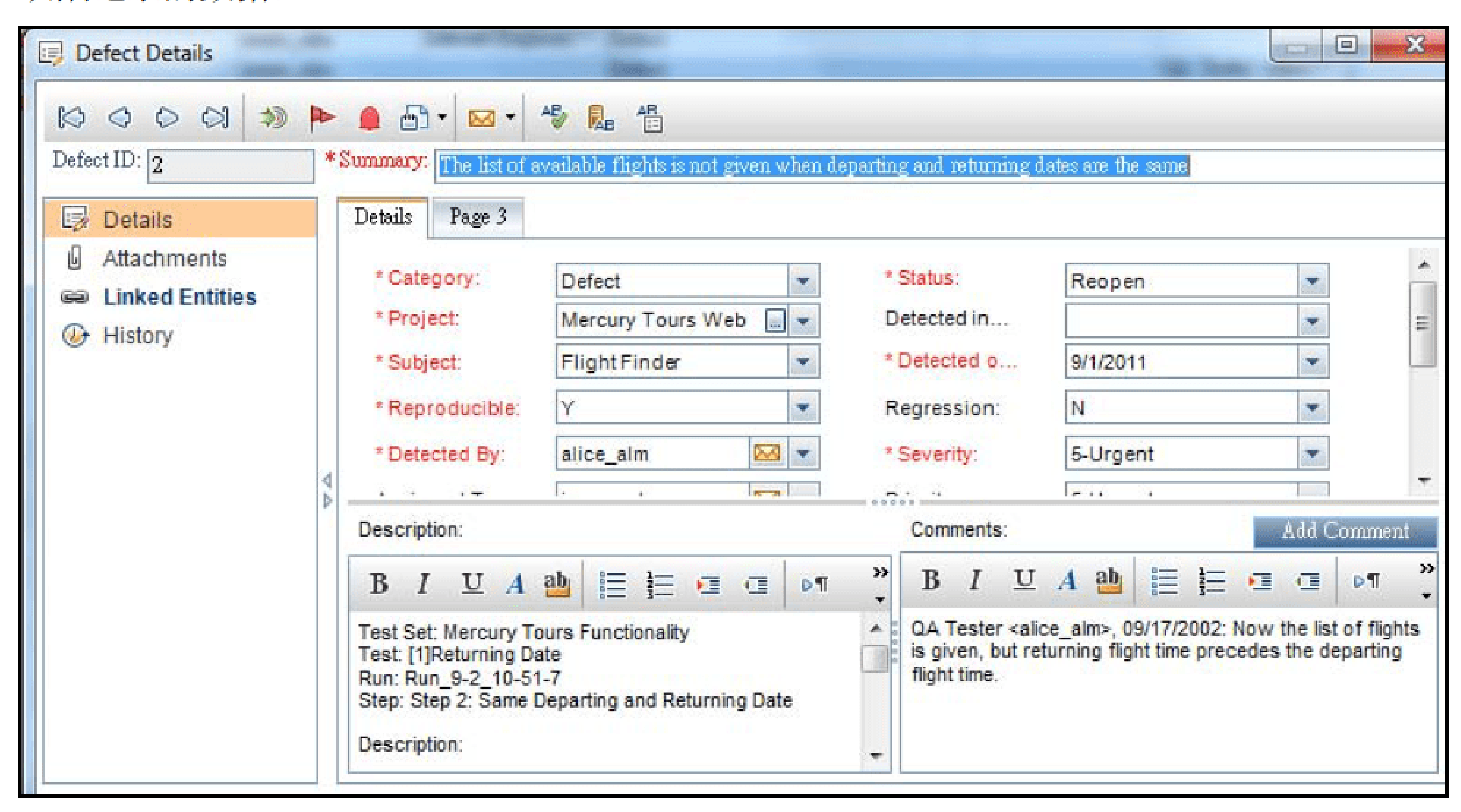


图 14-1　缺陷跟踪

管理和追踪缺陷在测试过程中是很关键的一步，因为这个过程要花费大量的时间和金钱。如果缺陷没有被正确地追踪，那么在后期阶段测试团队需要付出一些额外的工作来追踪它们。这些额外的工作可能会引起项目的延迟交付，也可能会导致成本超支。ALM 提供的中枢缺陷追踪系统就是让测试和开发团队用来解决缺陷的。

ALM 中的缺陷模块提供了一个很完善的系统，其中包括登录、追踪、管理和分析应用程序的缺陷。要操作这个模块，在 ALM 的导航栏中单击缺陷菜单，缺陷模块窗口将列出发现的缺陷和它们的属性。如图 14-2 所示，ALM 的缺陷追踪工具为缺陷模块提供以下一些组件元素：

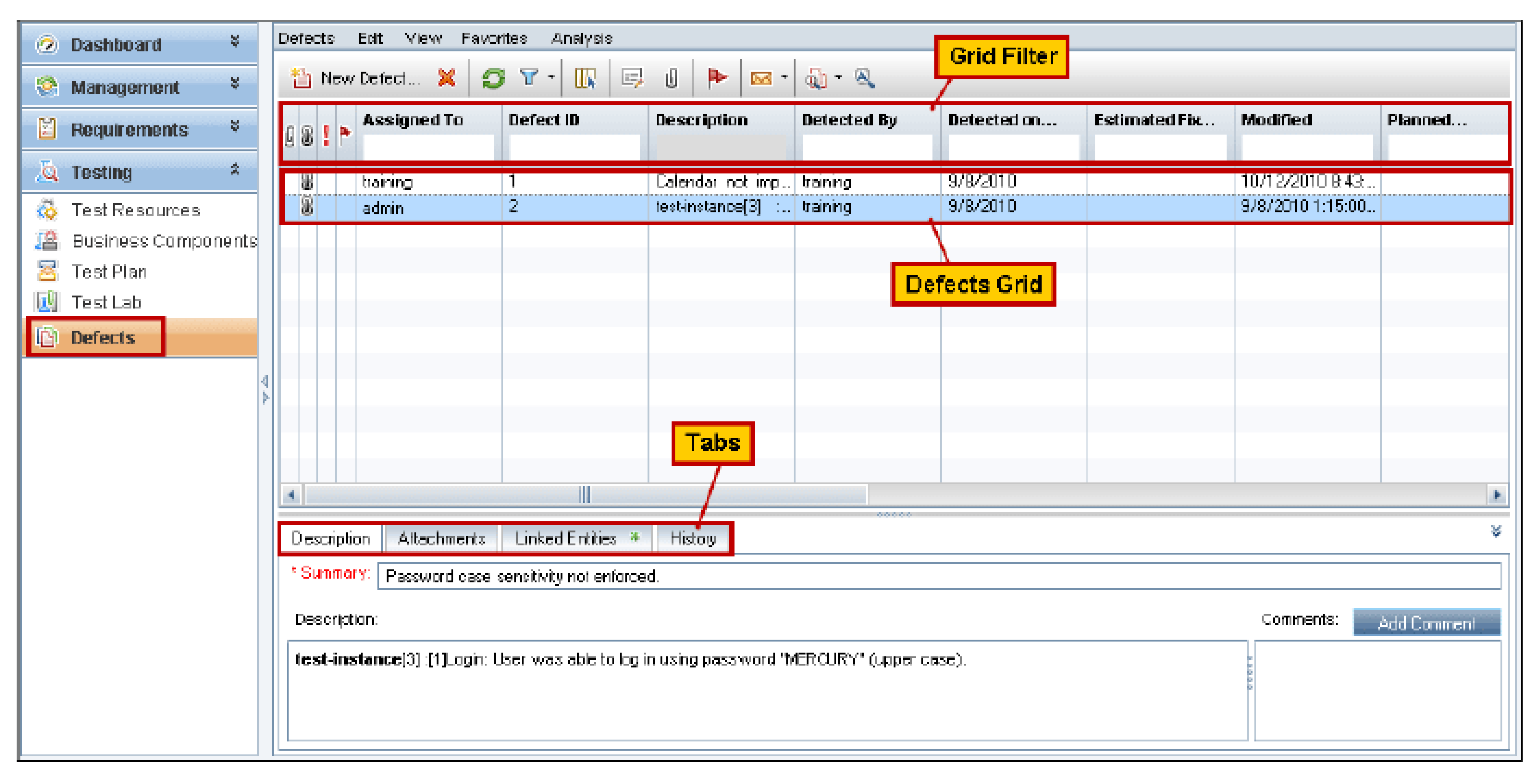


图 14-2　缺陷模块

1. Defects Grid：使用表格列出项目中的缺陷。

2. Grid Filter：可以在这里自定义标准来对缺陷网格中显示的数据进行筛选。

3. Description：提供一个文本框，可以为每个缺陷添加或修改注释。

4. Attachments：可以在这里附加文件、快照、URLs、系统信息和剪切图片来更好地说明缺陷。

5. Linked Entities：列出缺陷和与当前选择缺陷有关联的实体。

6. History：列出缺陷每一次所做出的修改记录，为缺陷提供一个追踪审查。

14.1.1　提交缺陷

报告缺陷就是为 ALM 项目创建一个新的被测试软件程序的缺陷(如图 14-3 所示)。

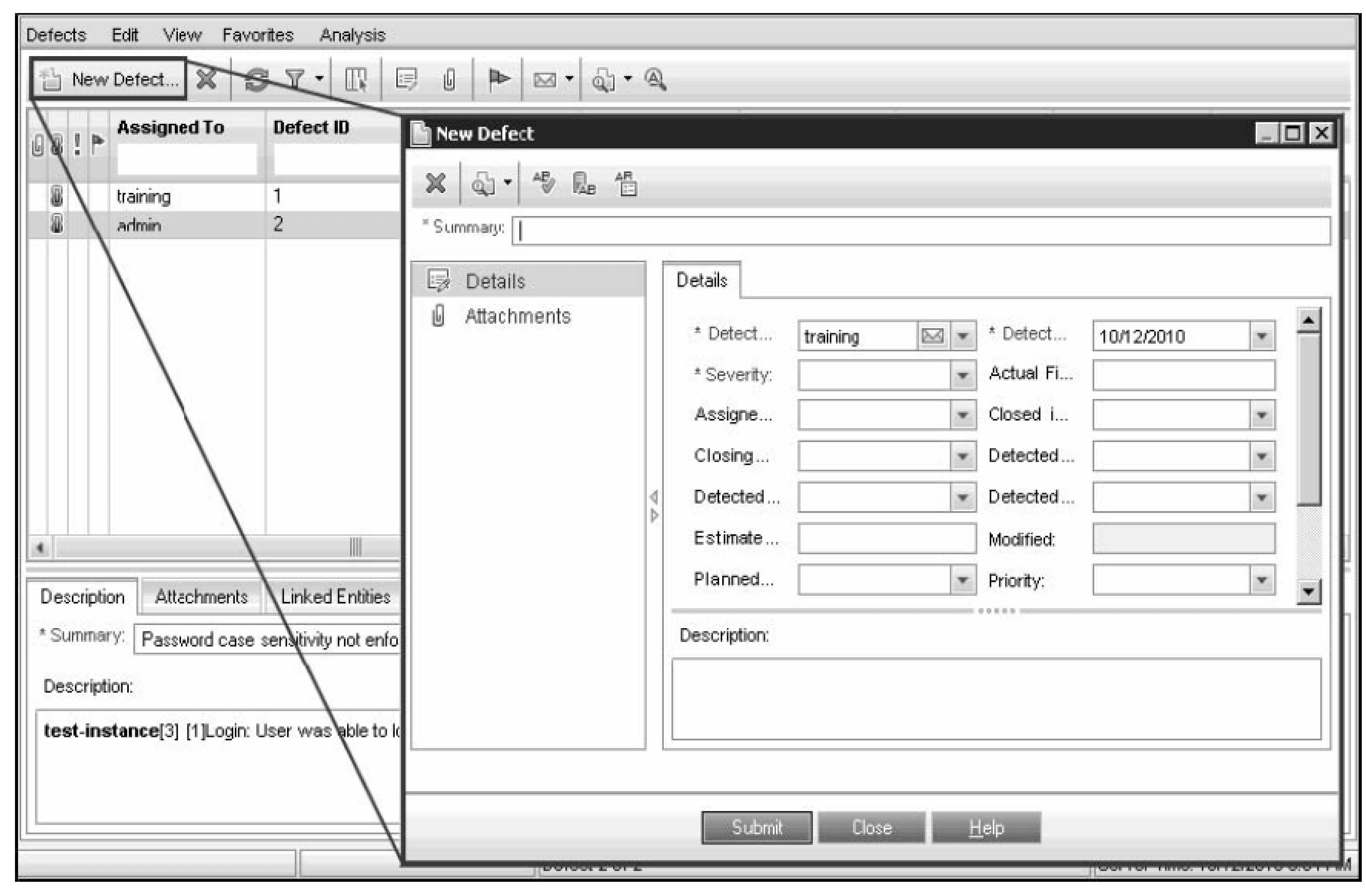


图 14-3　报告新缺陷

ALM 可以让你将任一测试过程中发现的缺陷记录下来。无论你是否定义需求、构建测试计划或执行测试运行，每个 ALM 模块都会提供一个常规工具来记录缺陷。例如，在需求和测试计划模块中提供的关联缺陷标签，你可以记录新的缺陷或者将已有的存在的缺陷与需求或测试关联起来。

持续追踪缺陷直到程序开发人员或测试人员确定缺陷已经被解决。缺陷记录是用来通知应用开发和质量保证团队的成员其他人发现了新的缺陷。当你监控的缺陷被修复后，你需要在 ALM 项目中更新信息。

1. 创建缺陷

1) 在 ALM 导航栏，选中 Defects。
2) 单击 Defects Grid 工具栏中的按钮，填写“新建缺陷”对话框中的字段。

2. 导入缺陷

除了直接在 ALM 中创建缺陷之外，还可以从 Word 或 Excel 中导入缺陷到你的 ALM 项目。要导入缺陷，需要先安装相应的插件。

3. 关联缺陷到其他实体

可以将缺陷与需求、测试、测试集、测试接口、运行、运行步骤和其他缺陷关联起来。

4. 搜索缺陷

在缺陷模块中，你可以从以下情况搜索缺陷：
1) Defects Grid，例如寻找与特定问题相关的缺陷，或修复一个特定的问题。
2) 在 New Defect 对话框中，例如避免创建重复的缺陷。

5. 分配、修复和更新缺陷

定期检查项目中的缺陷。决定修复哪些缺陷，并修改缺陷的状态，计划发布版本号、目标释放、计划目标周期和一些其他的相应字段。在缺陷模块中，双击缺陷网格中的一个缺陷。填写“缺陷详细”对话框中的字段。在其他任务中，你可以：

1) 把缺陷的状态改成未修复。将缺陷分配给开发团队的成员。

2) 在目标释放一栏，将打开的缺陷分配成释放代表目标被修复。在目标周期一栏，将缺陷分配到周期则表示目标已经关闭。在计划发布版本号一栏，为打开的缺陷选择一个计划被修复的版本号。在计划周期一栏，为缺陷选择一个计划被关闭的周期。

3) 修复打开的缺陷。其中包括识别引起辨别缺陷的原因，修复并重建程序，重新运行测试。如果一个缺陷没有再次发生，则可以将缺陷的状态改为 Closed。如果缺陷再次发生，则可以将状态指定为 Open。当一个缺陷被修复后，可以将它的状态指定为 Fixed。

6. 对缺陷进行分析

通过下面一种方式生成报告和图表来对缺陷进行分析：
1) 用图表显示缺陷数据，在缺陷模块菜单中，选择 Analysis | Graphs。

2) 创建一个缺陷数据报告，在缺陷模块菜单中，选择 Analysis | Report。

14.1.2　搜索重复缺陷

作为一个好的实践，在录入一个缺陷之前你需要先查看是否存在和你描述同一个问题的缺陷，这样可以避免重复提交和降低测试品质。利用工具栏中的网格过滤或查找缺陷，你可以检查是否存在和你想要提交的内容重复的缺陷。从缺陷模块中记录一个缺陷的步骤如下：

(1) 在缺陷模块中单击 New Defect，将出现 New Defect 对话框，对话框包括 Details 栏和 ALM 管理员根据项目而定制的许多选项卡。

(2) 使用恰当的信息来描述缺陷，除了填写新建对话框中的资料栏外，还可以添加附件来进一步详细说明缺陷。

(3) 单击 Submit 按钮，将缺陷保存到数据库。如果需要记录另一个缺陷，只需要刷新 New Defect 对话框。如果不需要，则单击 Close 按钮来关闭新建缺陷对话框。

记录缺陷后，可以用网格筛选来整理缺陷网格，并只显示你需要处理的缺陷。可以采用下面两种方式来使用网格过滤器：

1. 使用网格过滤器中每个字段标题下面的输入框入口，来选择过滤缺陷网格中数据的条件标准。

1) 单击字段标题下面的输入框入口，出现浏览按钮；

2) 单击浏览按钮，将出现 Select Filter Conditions 窗口。选择一个筛选条件并单击 OK。

2. 使用过滤对话框来设置筛选条件

1) 在工具栏中，单击设置筛选/排序按钮来打开筛选对话框。可以为筛选对话框中的每个字段名都设置筛选条件；

2) 单击 Group 选项卡。

注意：*如果你需要的字段并没有在缺陷网格的列中出现，那么在工具栏中单击选择列按钮，然后添加你需要的列。*

3) 从 Visible Columns 列表中，选择一个字段来对缺陷分组。

如果要清除存在的筛选规则，在工具栏中单击 Clear Filter/Sort 按钮。为了避免重复提交缺陷，测试人员可搜索是否有相似的缺陷(如图 14-4 所示)。从缺陷网格中，单击寻找相似缺陷按钮并选择相似文本，将会弹出相似缺陷面板。假如你想搜索关键字 Password 并查看所有和 Password 问题相关的缺陷，如果你只想看限制你的结果为 10%的搜索结果，搜索将返回下面的结果(如图 14-5 和图 14-6 所示)。



图 14-4　搜索重复缺陷

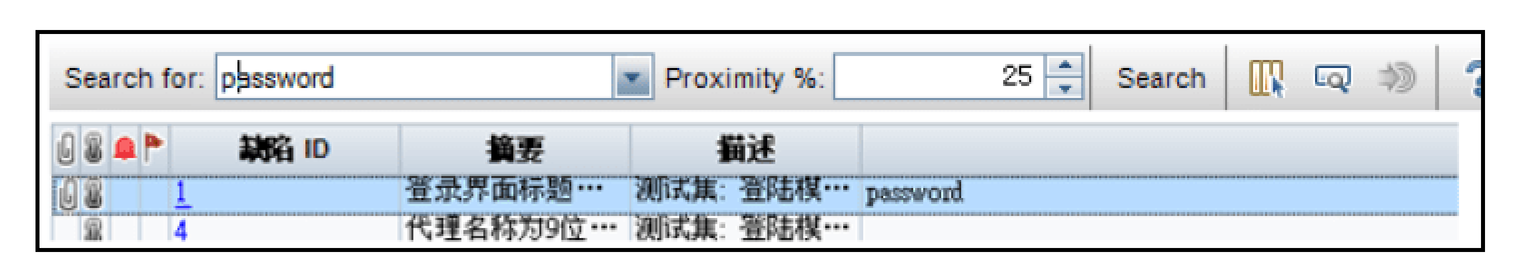


图 14-5　搜索关键字“Password”

缺陷 ID	摘要	描述	
1	登录界面标题…	测试集：登陆模…	password
4	代理名称为9位…	测试集：登陆模…	
2	代理名称由纯…	测试集：登陆模…	
3	代理名称由数…	测试集：登陆模…	password
8	点击帮助按钮…	测试集：登陆模…	
5	代理名称为特…	测试集：登陆模…	
7	密码输入框无…	测试集：登陆模…	
6	密码小于4位登…	测试集：登陆模…	
9	密码由纯数字…	测试集：登陆模…	
10	密码由字母和…	测试集：登陆模…	

图 14-6　搜索关键字“Password”返回的结果

14.1.3　缺陷状态

可以给缺陷分配一个状态来标记它在周期中所处的阶段。在每一个项目中可以用缺陷网格或详细缺陷对话框中的状态来为每个项目指定相应的状态。这个字段使得追踪缺陷更加容易。例如，你可以用状态作为筛选标准来运行查询和和生成报告的筛选标准。图 14-7 显示的就是如何使用系统定义的状态来显示一个缺陷在周期中的不同阶段。

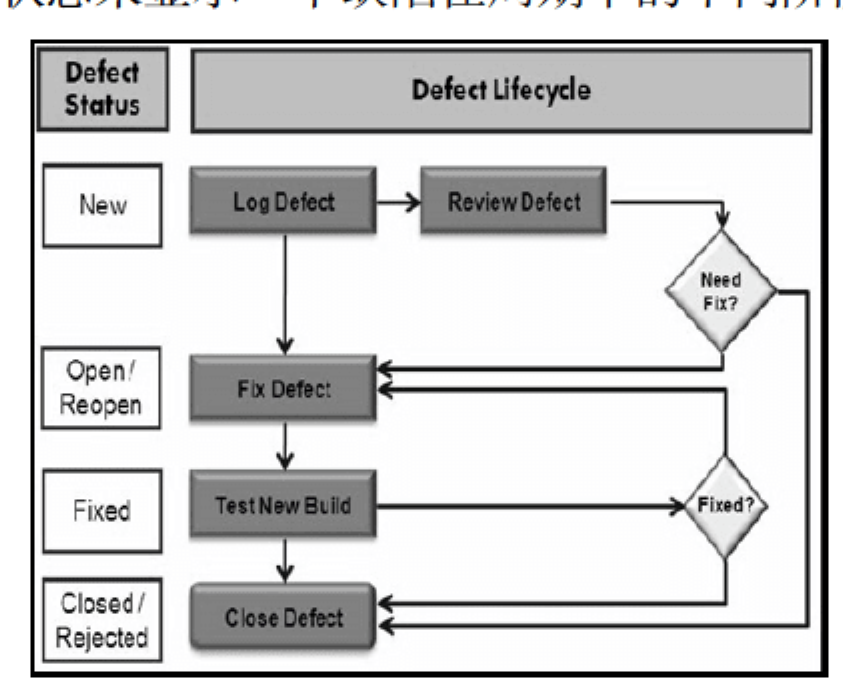


图 14-7　缺陷生命周期

默认的缺陷状态有：

1. New：表明缺陷刚提交。

2. Open：表明缺陷已经指派给开发团队去检查。

3. Reopen：表明测试团队重新打开开发团队关闭的缺陷。

4. Fixed：表明开发团队已经将缺陷修复，但是在等待测试团队的确认。

5. Closed：表明测试团队已经核查，缺陷被修复。

6. Rejected：表明开发人员不接受这个缺陷。但是开发人员需要为拒绝缺陷提供一个论据。

注意：一般来说，你所在的组织和你所开发的应用程序会有不同的需求。因此，你可以

用状态标签来区别你的每个项目。在 ALM 的项目计划和订制课程中使用状态标签。另外，每一个项目都会有不同的客户化的标准来定义哪些用户可以修改哪些状态，哪些条件下缺陷的哪些状态可以被用户修改。举例：项目里的缺陷必须是从新建状态开始，缺陷没有被修复是不能被关闭的，只有质量保证团队才能把缺陷状态置为关闭状态。

14.2　缺陷与其他实体关联

一个缺陷可以直接附加到任何实体(缺陷、运行步骤、运行、测试实例、测试集、测试集文件夹、测试和需求)。可以手动地向右间接关联所有实体。例如，如果附加到测试实例的缺陷可以附加到测试集(如图 14-8 所示)。如果缺陷被关联到测试，那么它会自动关联到包含在测试里的需求文档。

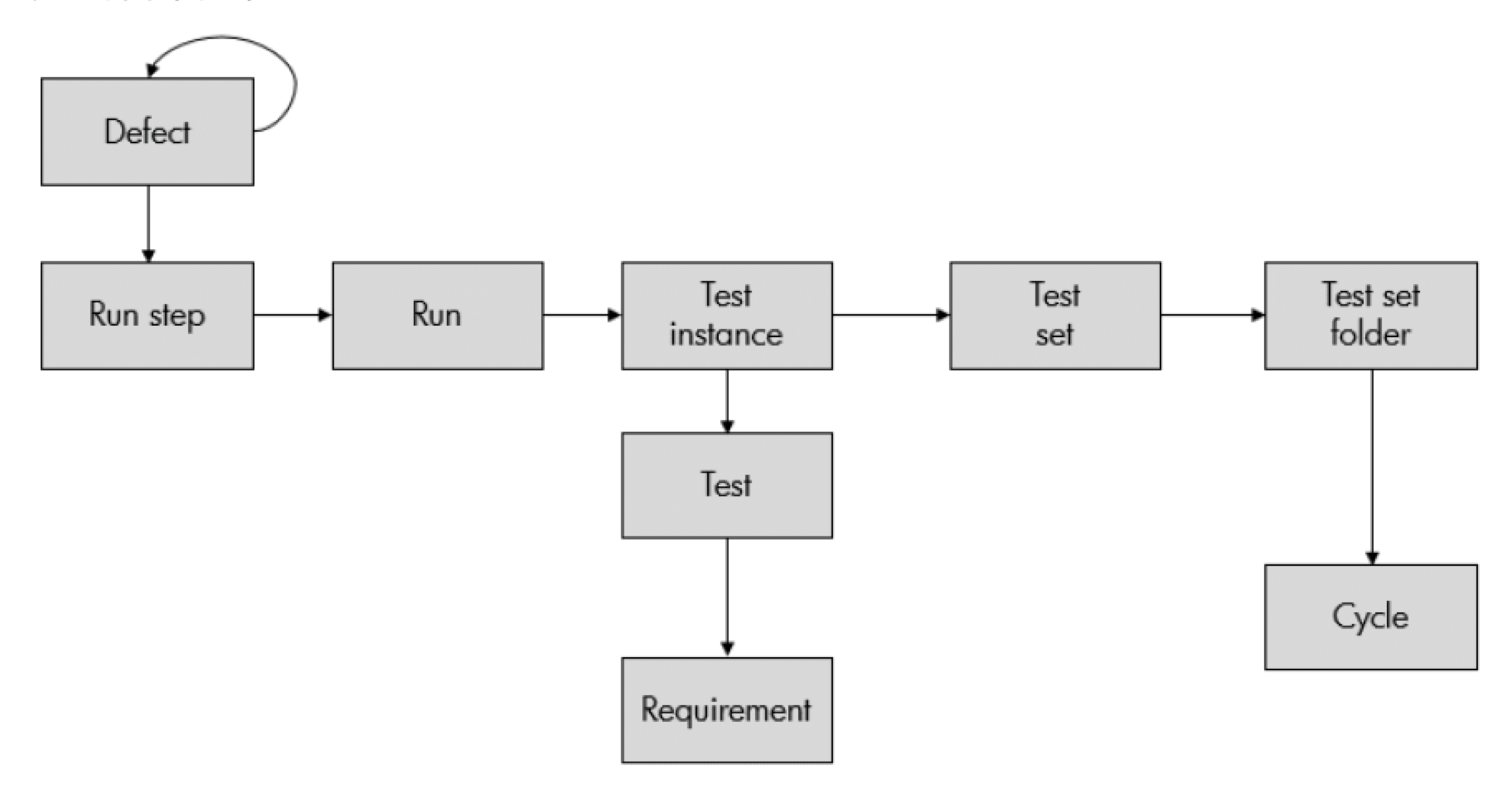


图 14-8　关联缺陷与其他实体

所以，在测试运行时就直接把缺陷附加到运行步骤是一个好习惯，其他的都是间接地(虽然也可以后期手动添加)。如果缺陷与测试的实例的关系被删除，但是缺陷与测试的关系依然存在，因此它们还会在新的测试实例中展现。

14.2.1　缺陷-需求关系

为了确保测试过程的一致性，可以将缺陷与需求联系在一起。可以将已存在的缺陷或新建的缺陷与需求关联。缺陷-需求关联有下面一些特征：

1. 根据缺陷-需求关联，你可以用缺陷的状态来检验与之相关联的需求是否满足。例如，如果存在一个并不满足的需求，但是有一个缺陷是因为这个需求的所有测试都运行而上报的。

2. 一个需求可以和多个缺陷关联。例如：假设你为你的业务流程定义了一个需求 R_01，并且你确定了一个与这个需求有联系的缺陷，你将缺陷命名为 Defect_01，在整个测试过程中 Defect_01 与 R_01 关联在一起。这意味着当 R_01 与一个测试关联时，Defect_01 仍然与 R-01

关联。

14.2.2　添加缺陷到需求

如图 14-9 所示，将新的缺陷添加到需求的步骤如下：

(1) 在 ALM 的导航栏，单击 Requirements 菜单。

(2) 在 Requirements 模块，从菜单栏中选择 View | Requirement Details。

(3) 从 Requirements Tree 中，选择你想要添加缺陷的需求。

(4) 在右边的面板中，单击 Linked Defect 标签。

(5) 在关联缺陷工具栏中，单击 Add and Linked Defect 按钮，将出现 New Defect 对话框。

(6) 在需求字段填写相应的信息。单击 OK 来添加缺陷。

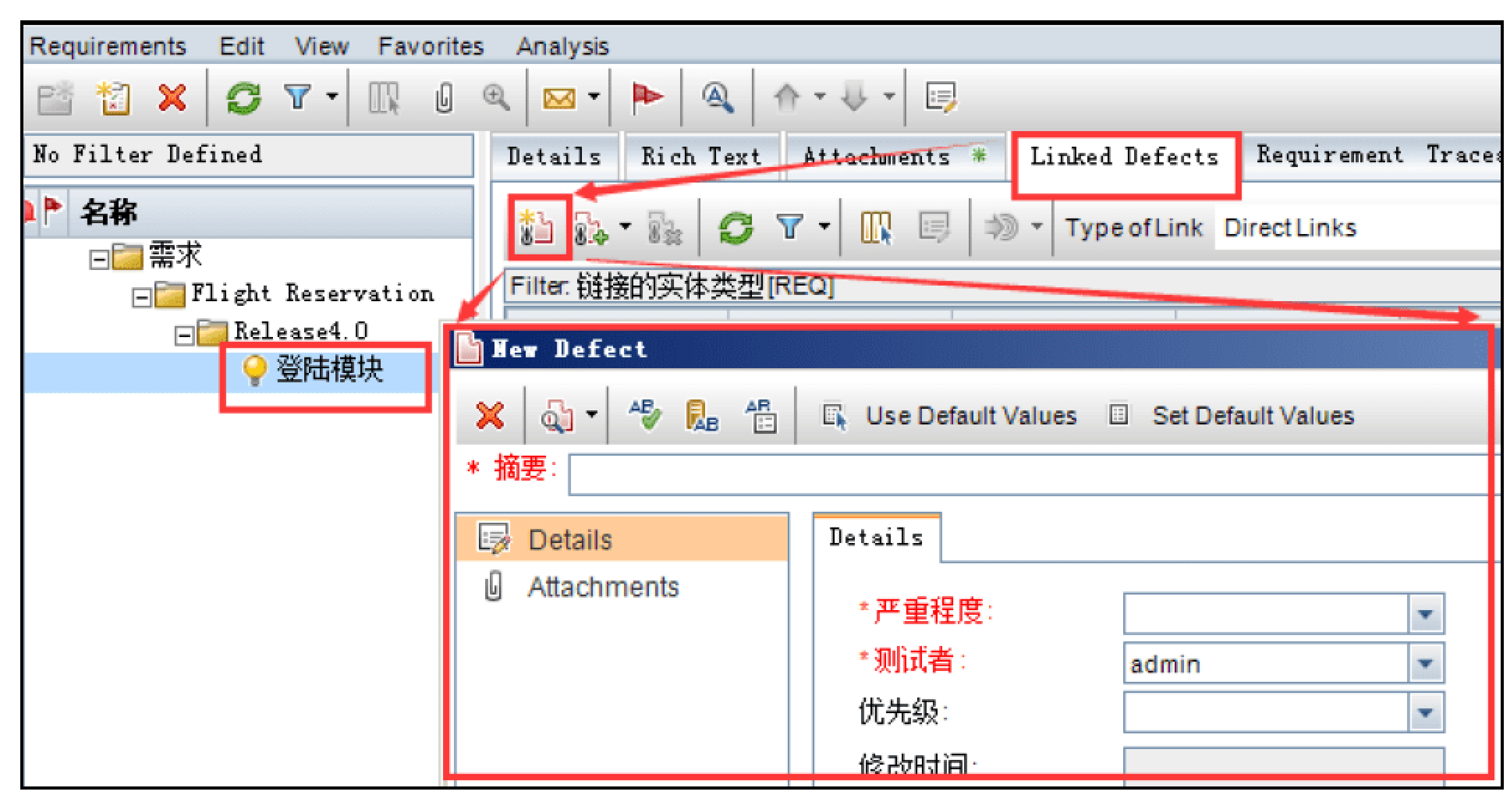


图 14-9　添加一个缺陷到需求

14.2.3　关联存在的缺陷和需求

如要将已存在的缺陷与需求关联，需要按以下步骤：

(1) 在 Requirement Details 视图中，从需求树中选择想要与缺陷关联的需求。

(2) 在关联需求页面的工具栏中，单击 Linked Existing Defect 箭头，单击 Select。将出现缺陷关联对话框。

(3) 在缺陷关联对话框中，选中一个缺陷。

(4) 单击 Link，将选中的缺陷与需求关联。

图 14-10 显示将已存在的缺陷与需求关联：

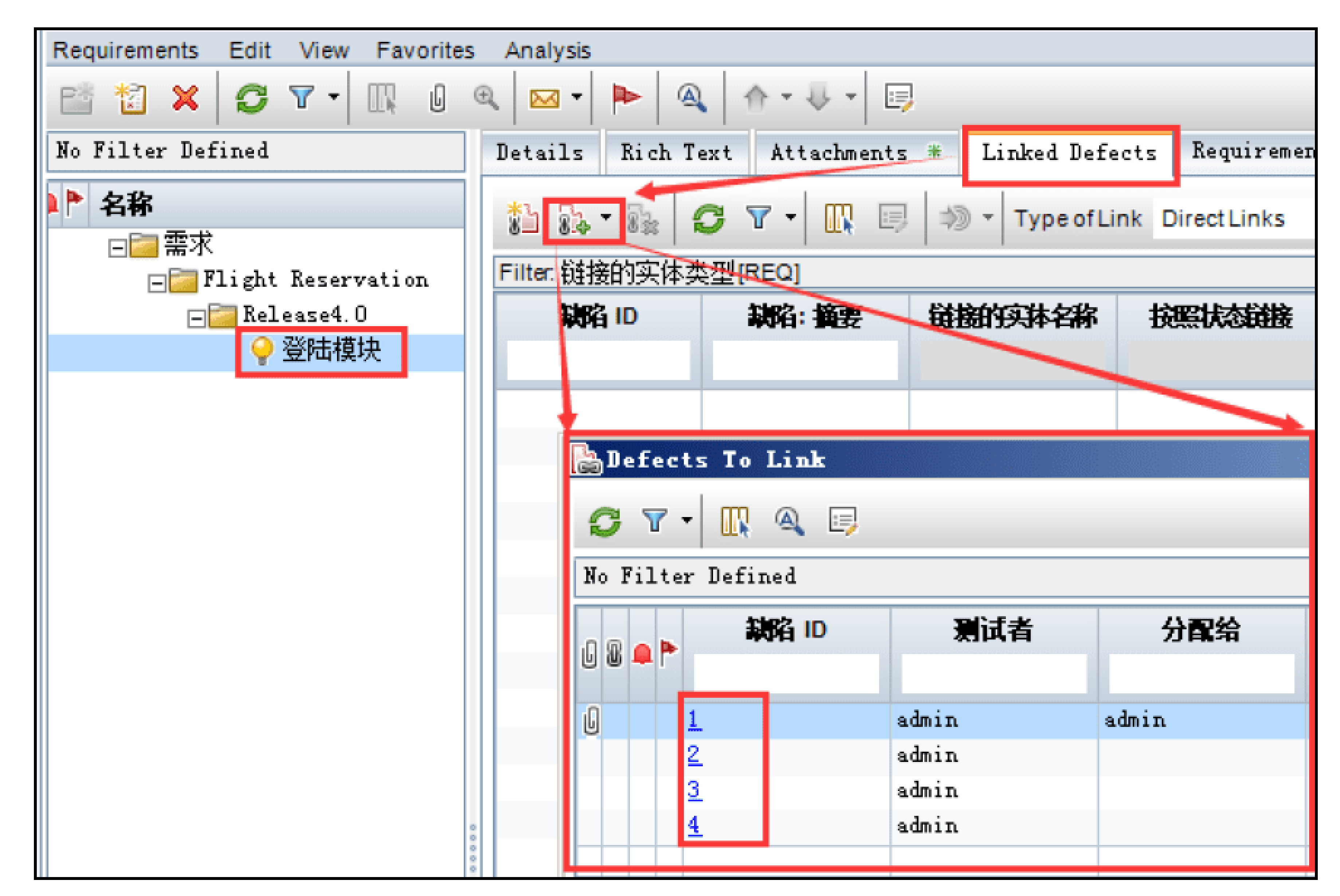


图 14-10　显示将已存在的缺陷与需求关联

14.2.4　缺陷-测试关系

可以将缺陷与测试关联来确保它们在整个测试过程中的可追踪性。这是缺陷和测试之间的一个直接关联。缺陷也可以通过其他实体与测试间接地关联，例如测试实例、测试运行或测试步骤。在缺陷与测试关联后，将更容易追踪测试的所有用例。这些测试用例可以在同一个测试集也可以在不同的测试集。

假设有一个测试的名称为 Test_01，可以假设 R_01 是我们在缺陷-需求关系主题下创建的需求，它和 Test_01 相关联。因此，已经和 R_01 关联的 Defect_01 现在也与 Test_01 相关联。现在 Test_01 中有三个实例在不同的测试集。因此，无论这个测试的实例在三个测试集中的任一个中执行，Defect_01 都将会直接关联到指定的测试实例。

缺陷-测试关联有以下特征：

1. 测试计划模块中的测试可以与缺陷模块中记录的缺陷相关联，这种关联是可以让你使用缺陷的状态作为检验是否及何时应该运行测试的基本根据。另外这些测试包含的需求也可以手动与它们相应的缺陷相关联。

例如，你想只在缺陷状态为已关闭时才运行测试，这意味着开发团队已经修复好缺陷，缺陷处于 Fixed 状态。这表示确保开发与测试团队之间已经达成了一个沟通的协议，从而将修复的时间减到最小。

2. 手动测试运行时记录的缺陷可以手动与特定的测试运行相关联。

3. 一个测试可以与多个缺陷相关联。

14.2.5 添加缺陷到测试

添加缺陷到测试的具体操作方法如下：

(1) 在 ALM 导航栏中，单击 Test Plan 菜单。

(2) 在左边的面板中，从 Test Plan Tree 中，选中你想要添加缺陷的测试。

(3) 在右边的面板中，单击 Linked Defects 标签。

(4) 在关联缺陷页面的工具栏中，单击 Add and Link Defect 按钮，将出现 New Defect 对话框。

(5) 在 New Defect 对话框中，在必填的字段处填写相应的信息，然后单击 OK 来添加缺陷。

如需要添加一个已存在的缺陷到测试实例，在关联缺陷面板工具栏中，单击 Linked Existing Defect 箭头。可以填写缺陷 ID 或单击 Select 从缺陷网格中选择缺陷(如图 14-11 所示)。

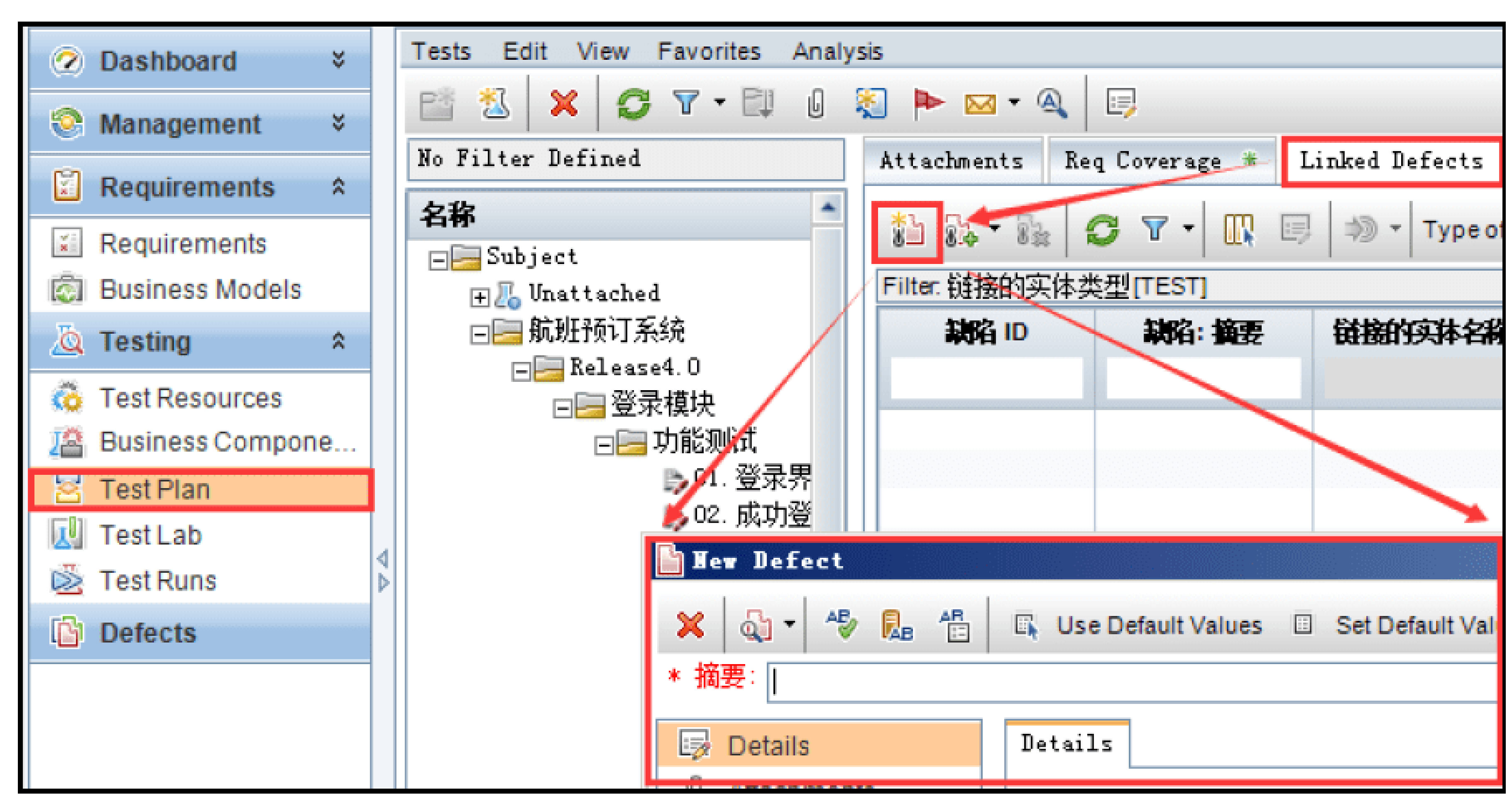


图 14-11 缺陷-测试关联

14.2.6 缺陷-测试实例关系

如果一个测试实例在多个测试集中，则缺陷可以与测试集相关联来确保易于追踪。

缺陷-测试实例关联有下面的特征：

1. 测试实验模块中的测试实例可以和缺陷模块中记录的缺陷相关联，这可以帮助你判断出那些没有正确运行的测试实例是否正确工作。

2. 当一个缺陷记录到一个测试实例时，因为对应测试用例的缘故，它也会自动地记录到一个测试，这个测试是测试实例的父节点根源。

3. 如果缺陷测试实例的关系被删除，但是缺陷测试的关系仍然存在。这保证了这个缺陷会被报告到该测试的其他的新实例中。

将缺陷与一个测试实例关联的步骤如下：

(1) 操作 Test Instance Details 对话框。

(2) 将缺陷与测试用例关联。

14.3　详细操作

如图 14-12 所示，操作 Test Instance Details 对话框的步骤如下：

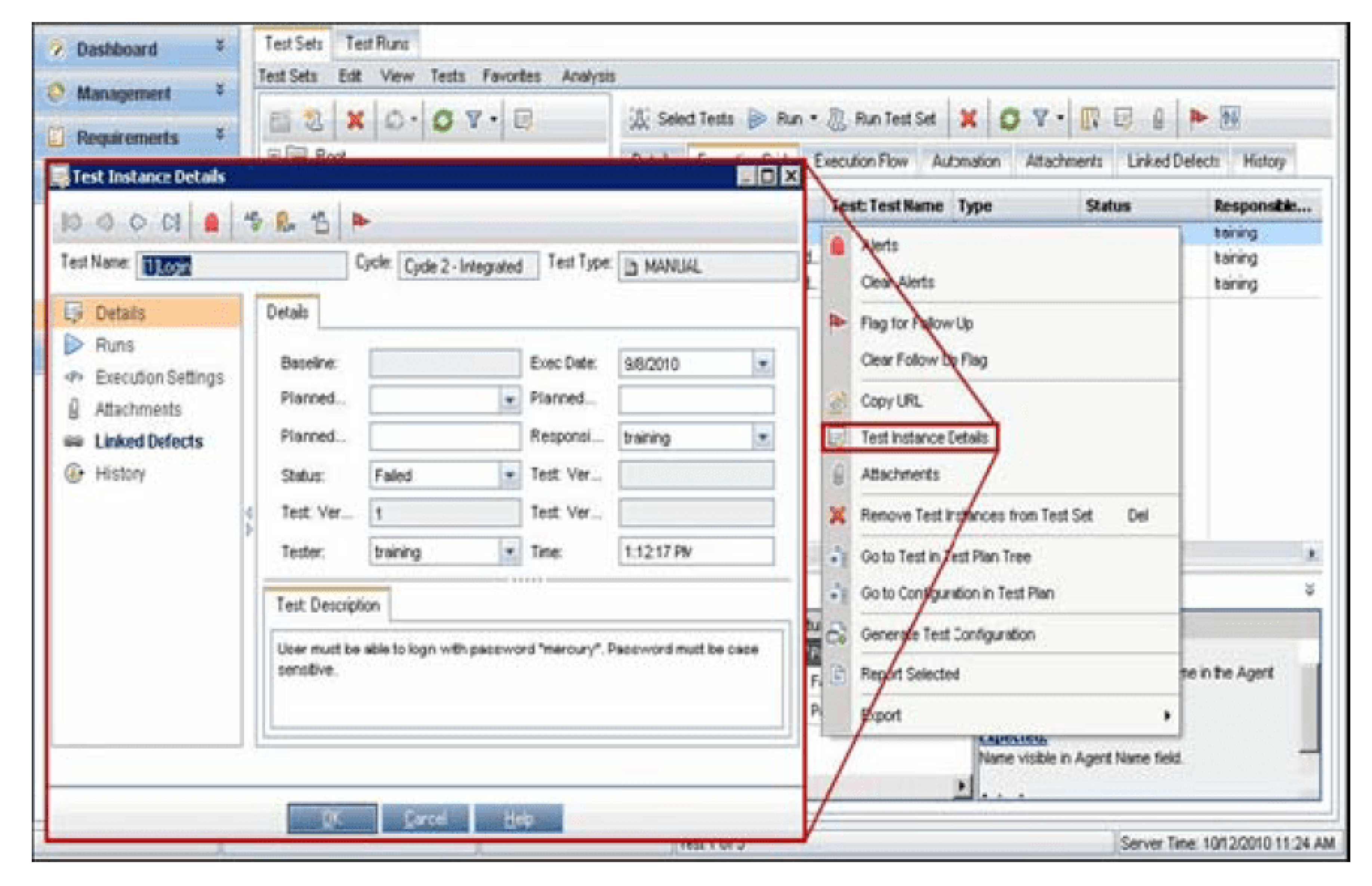


图 14-12　Test Instance Details 对话框

(1) 在 ALM 导航栏中，单击 Test Lab 图表；

(2) 在左边面板中，从 Test Set 中选中一个测试集；

(3) 在右边面板中，单击 Execution Grid 标签；

(4) 在 Execution Grid 标签中，右击一个测试，选择 Test Instance Details。将出现 Test Instance Details 对话框。

14.4　关联缺陷和测试实例

操作完 Test Instance Details 对话框后，可以将缺陷与测试实例关联(如图 14-13 所示)。

关联缺陷与测试实例的步骤如下：

(1) 在 Test Instance Details 对话框的导航栏中，单击 Linked Defects；

(2) 在工具栏中，单击 Linked Defects 按钮，将出现 New Defect 对话框；

(3) 在 New Defect 对话框中，输入必填的信息，单击 OK 添加缺陷到测试用例。

注意：如需将已存在的缺陷与测试用例关联，单击 Linked Existing Defect 箭头。你可以填写缺陷 ID 或单击 Select 从缺陷网格中选择一个缺陷。

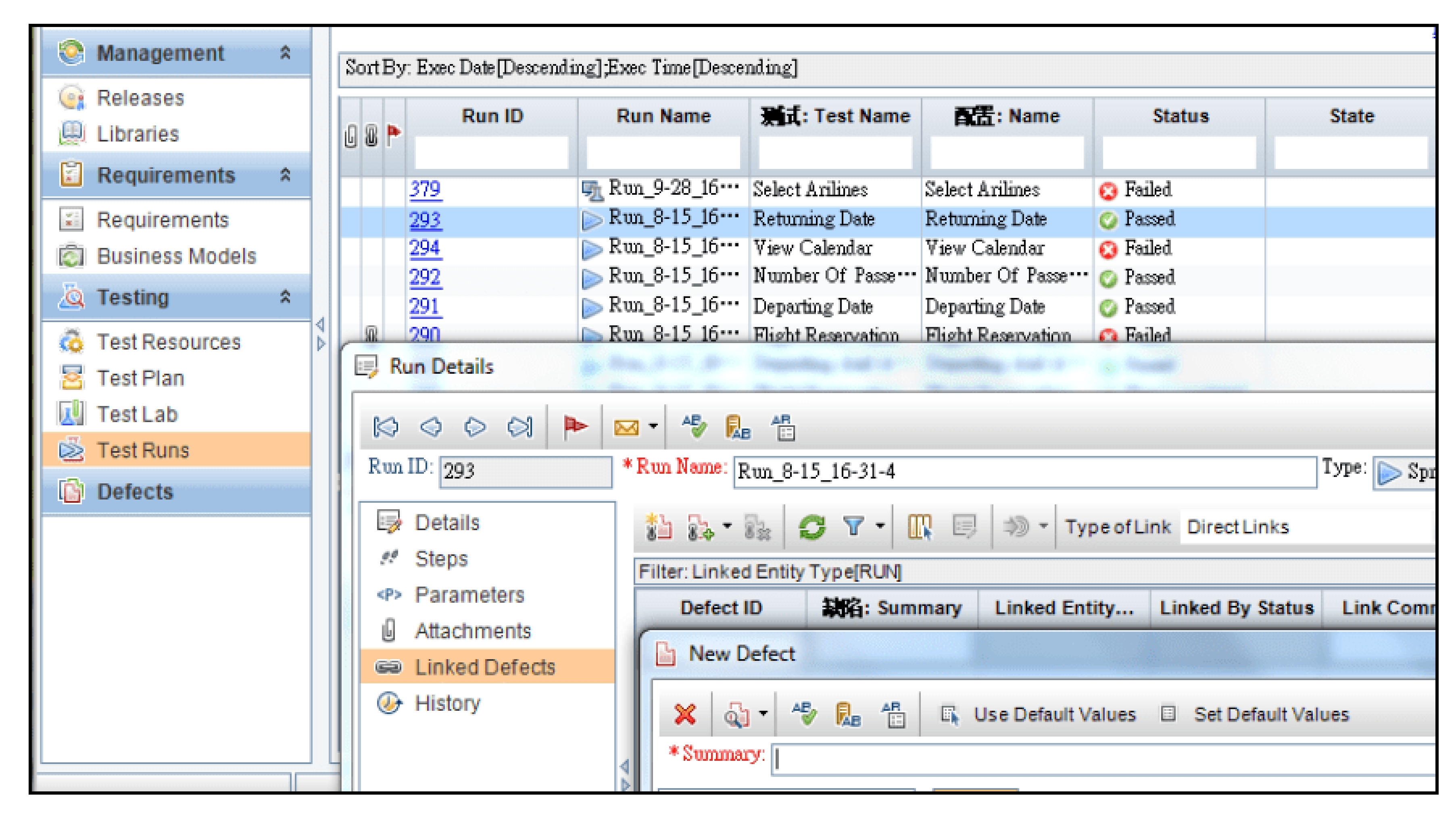


图 14-13　关联缺陷和测试实例

14.5　执行测试时记录缺陷

在手动测试运行时记录一个缺陷(如图 14-14 所示)的步骤如下：

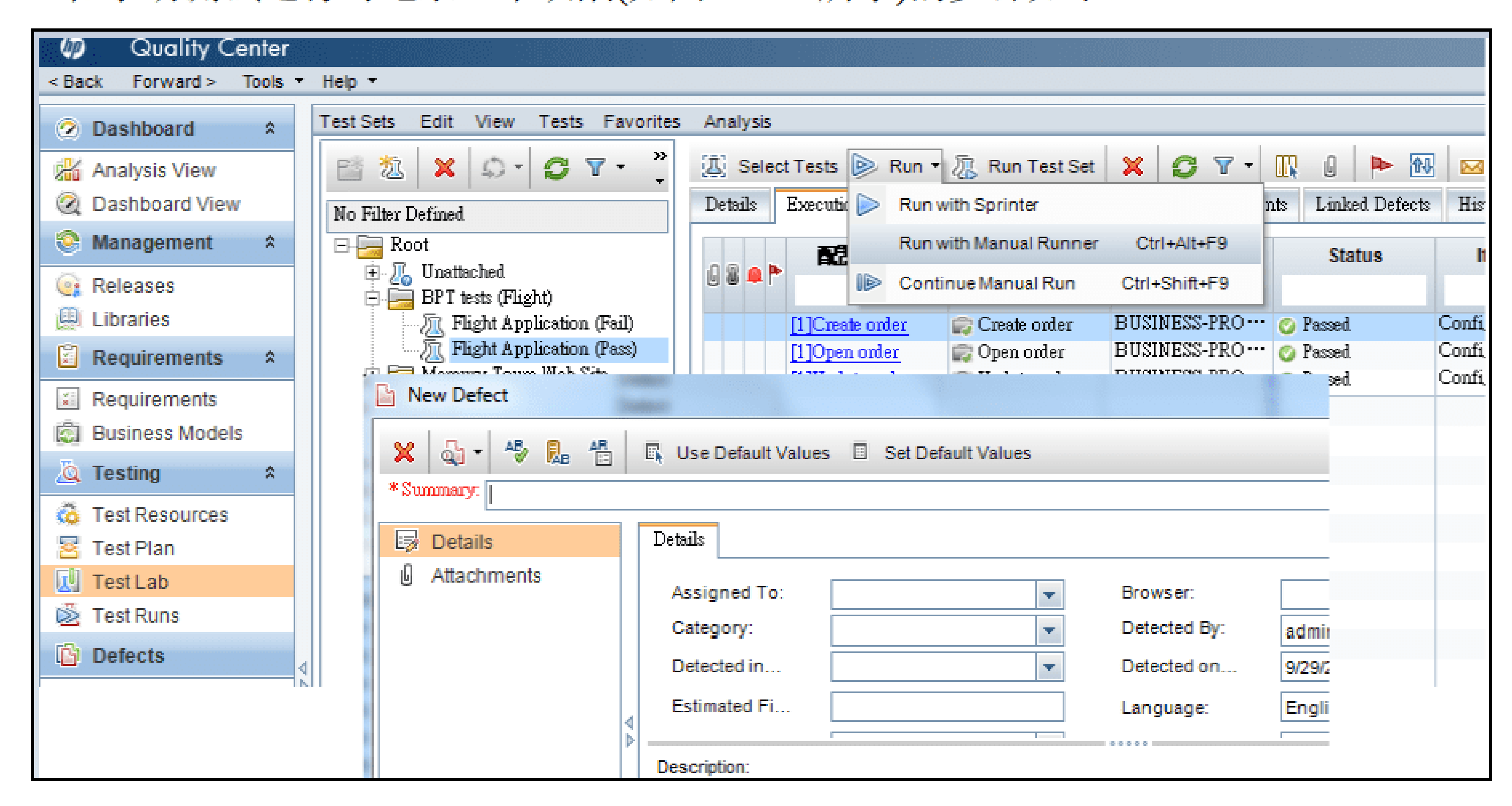


图 14-14　执行测试时记录缺陷

(1) 在执行表格页面中，选中一个测试，单击 Run 箭头来选中手动运行。将出现 Manual Runner: Test Set 对话框；

(2) 在 Manual Runner：Test Set 对话框中，单击 New Defect。将出现 New Defect 对话框；

注意： New Defect 对话框中的一些特定数据自动从当前测试运行中继承数据。例如，详细字段记录了测试集的名称、测试步骤和记录缺陷时运行的测试。当你需要检查和解决缺陷

时，这些特征提供了你需要的所有参考信息。

(3) 单击 OK 来保存缺陷到缺陷模块。

14.6　更新缺陷

定期的更新缺陷可以用来记录该缺陷的所有信息，并记录不同成员个体处理、修改缺陷状态时所作的各种决策。

更新缺陷的步骤如下：

(1) 在 ALM 的导航栏中，单击 Defect 菜单；

(2) 从缺陷网格中，双击一个缺陷，将打开 Defect Details 对话框。可以使用 Defect Details 对话框提供的工具条，对不同的视图进行切换。视图有：Details、Attachments、Linked Entities 和 History；

(3) 单击 Details 更新指定的字段数据；

(4) 单击 Attachments 来附加文件到缺陷。

例如，可以再添加一个测试过程中被测对象的缺陷报告文件或缺陷快照来帮助阐述问题。还可以附加一些运行测试的系统信息，帮助开发人员更好地重现问题：

(1) 单击 Linked Entities ，然后单击 Defects 标签将一个缺陷与另一个缺陷关联；

(2) 单击 Other 选项卡，将缺陷与其他实体关联，例如测试或测试集；

注意：可以使用 Link Comment 列来为关联添加评论。

(3) 单击 History 查看缺陷变更的历史记录。对于缺陷的每个变更，网格记录了每一次可以显示变更的数据、进行变更的用户、原始数据和新数据。

图说明了缺陷修改对话框。

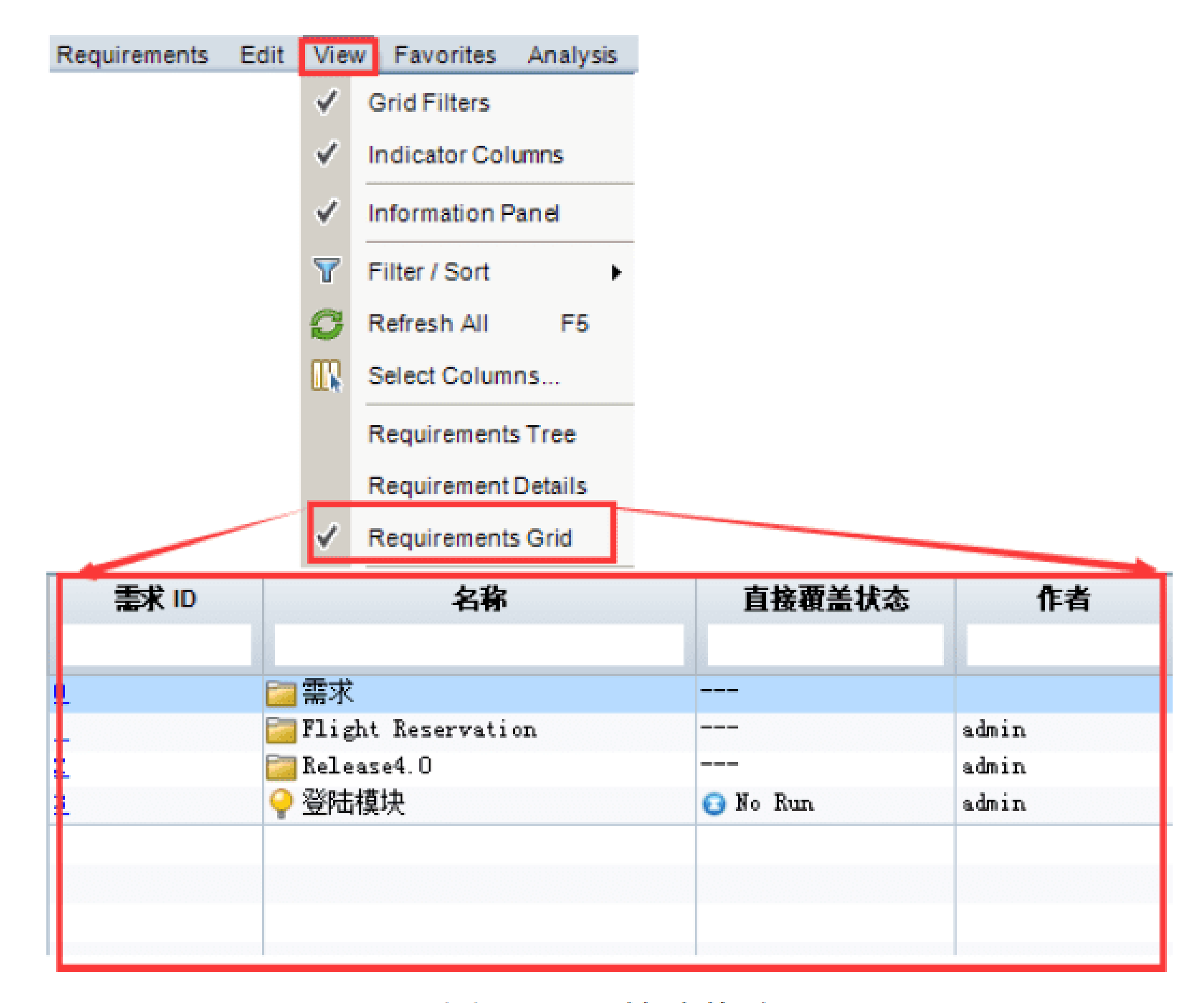


图 14-15　缺陷修改

14.7　关联缺陷/实体页面

可以在图 14-16 所示的这个页面定义和维护缺陷与缺陷、缺陷与实体的关联。

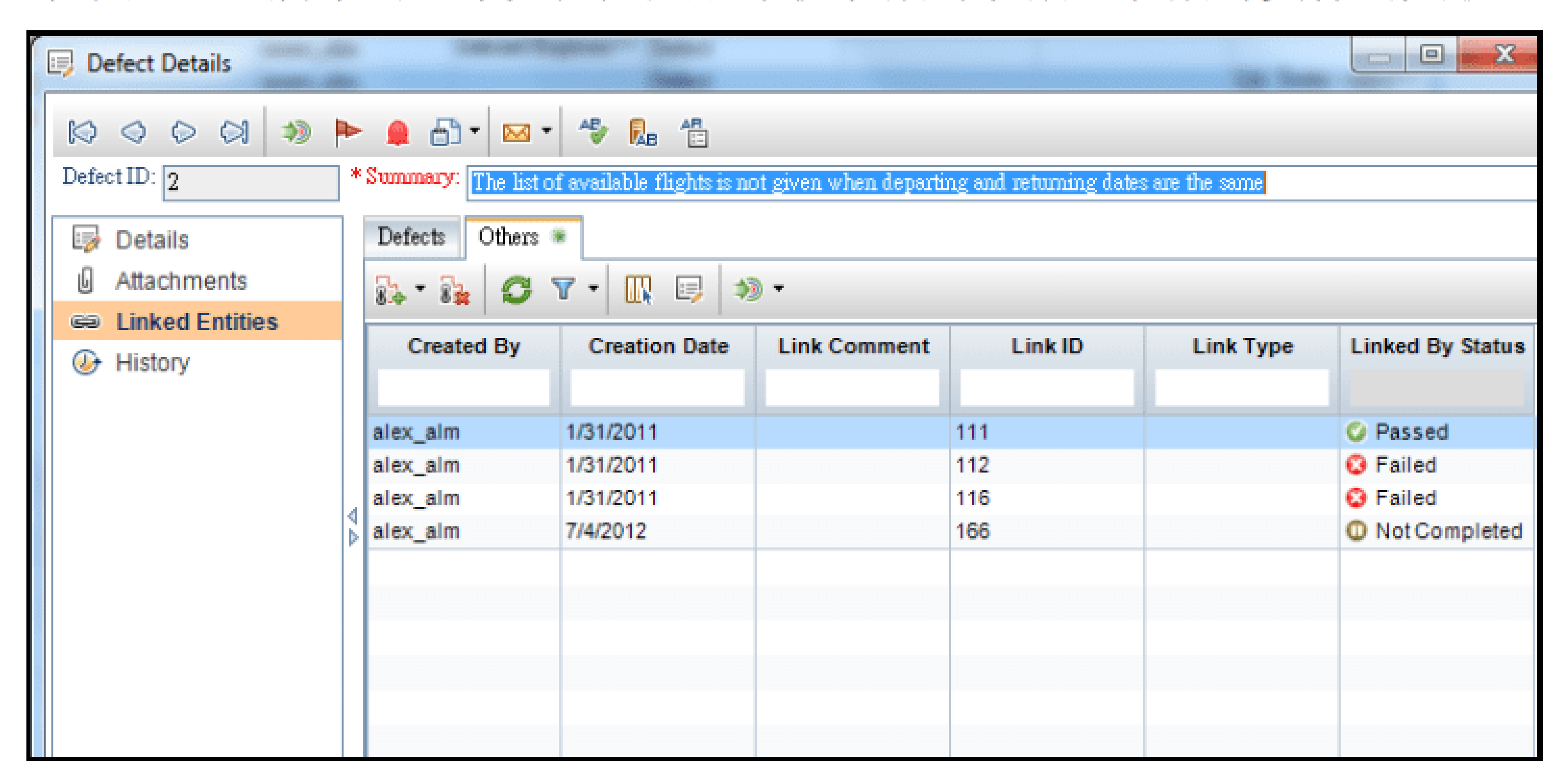


图 14-16　关联缺陷/实体

14.8　查看缺陷结果

如图 14-17 所示，可以从相应的缺陷中查看关联的实体，也可以从相应的实体来查看关联的缺陷。在网格视图中，当一个实体与缺陷关联时，ALM 会为实体添加一个关联缺陷的图标。

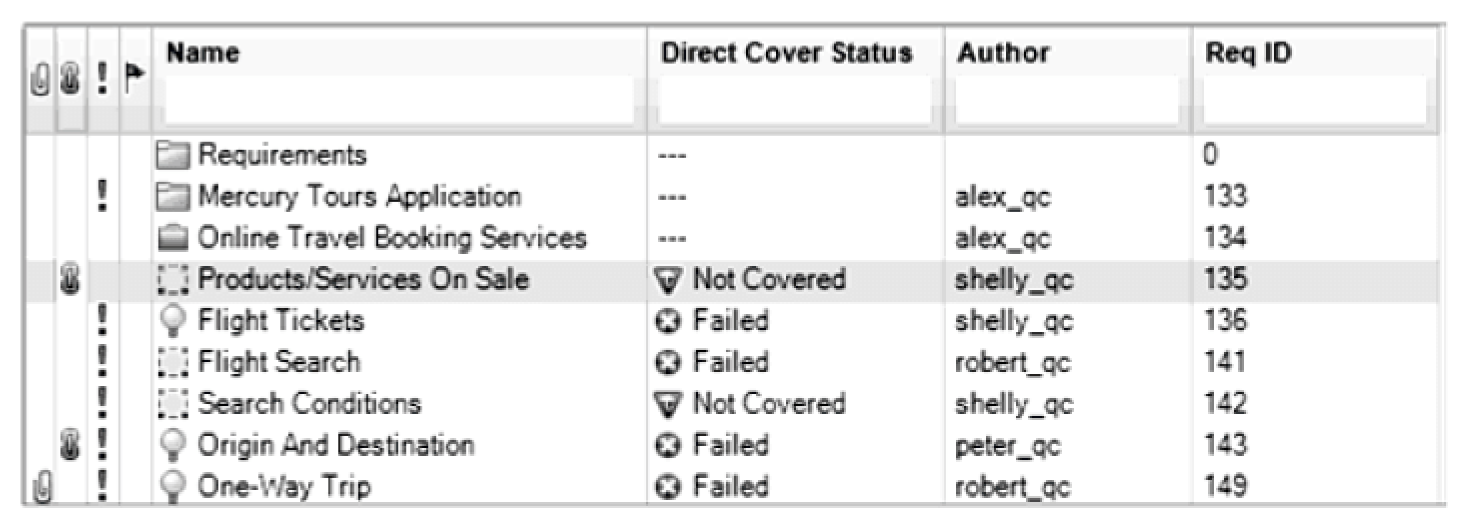


图 14-17　查看缺陷结果实例

习题与思考题

1. 可以直接与缺陷关联的不同实体有哪些？
2. 如何对缺陷的单个字段设置筛选器？
3. 书写一份高质量的缺陷报告要注意哪些地方？

练习：记录缺陷

缺陷的数据可以帮助你监控程序的质量。缺陷可以和其他实体关联，而报告可以确保缺陷能及时被分析并修复。可以使用下面的方式在 LOGIN 测试中关联一个缺陷。

1. 一个测试运行的是 LOGIN 测试。这个缺陷-测试运行关联表明了单个的测试运行 LOGIN 测试失败。

2. 创建预定功能需求：这个缺陷-需求关联意味着由于 LOGIN 测试有 14-15 一个缺陷，所以无法满足创建预定功能这个需求。

也可以记录一个缺陷来报告没有被实现的系统的功能。

这个练习中，需要执行下面的任务

第 1 部分：手动创建缺陷

第 2 部分：根据测试运行创建缺陷

第 3 部分：将缺陷与需求关联

第 4 部分：新建一个缺陷

(该练习的指导步骤请参见本章正文)

第15章　从Excel导出数据

15.1　导出数据

ALM 允许导出使用由第三方工具生成的文件内的测试数据。可以使用插件将数据从 Microsoft Word 和 Microsoft Excel 导出到 ALM。ALM 的 Excel 插件在 Excel 中安装宏命令，使你能够格式化数据并把数据从 Excel 导出到 ALM(下载插件页如图 15-1 所示)。

你能导出下列 Excel 工作表或 Word 文档数据到 ALM：

- 需求数据
- 测试计划数据
- 缺陷数据(仅 Excel)



图 15-1　ALM 下载插件页面

注意：本章节覆盖的范围仅仅为 Microsoft Excel 附件的安装和数据从 Excel 到 ALM 的导出。

15.2　导出数据步骤

在电脑中为 Excel 安装 ALM 插件后，就可以将数据导出到 ALM。导出数据的步骤如下：

(1) 安装 Excel 插件。

(2) 格式化 Excel 中的数据。

(3) 将数据导出到 ALM。

(4) 检查导出到 ALM 中的数据，进行必要的添加和调整。

15.2.1　安装 Excel 插件

安装 Excel 插件的步骤如下：

(1) 在 ALM 首页单击 Add-Ins Page 链接。弹出 ALM Add-Ins 页面(如图 15-2 所示)。

(2) 单击 More HP ALM Add-Ins。弹出 More HP ALM Add-Ins 页面。

(3) 在插件部分，单击 Microsoft 应用程序插件。弹出 Microsoft 应用程序插件页面。

(4) 单击 Microsoft Excel 链接，将插件下载到电脑。



图 15-2　Add-Ins 页面

15.2.2　确认插件安装成功

安装 Excel 插件之后，要验证插件安装是否成功。

验证 Excel 插件的安装：打开电脑中一个 Excel 工作表。在 Excel 2013 的菜单栏，选择 Tools。Tools 菜单中显示了 Export To HP ALM 选项。使用这一选项可以将数据从 Excel 导出到 ALM。在 Excel 2007，单击 Add-Ins 页面，页面中有 Export To HP ALM 按钮。

图 15-3 显示了 Excel 中用来导出数据到 ALM 的接口元素。

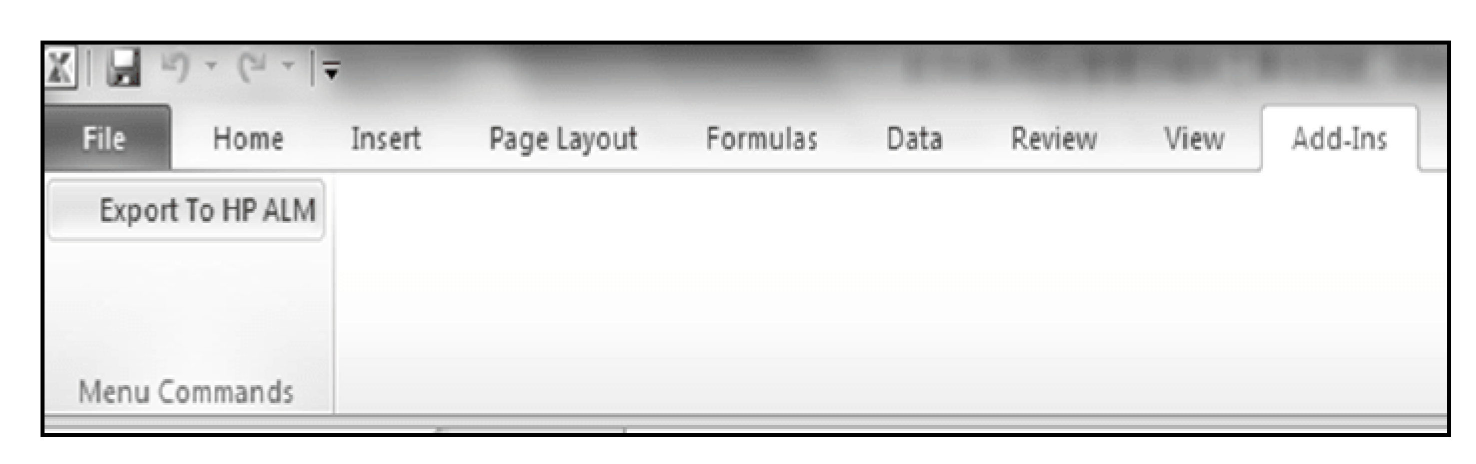


图 15-3　验证 Add-Ins 安装

15.2.3　格式化 Excel 文件

ALM 不提供在 Excel 中格式化数据的插件工具栏。因此，在将数据从 Excel 导出到 ALM 之前，需要在 Excel 中手动地格式化数据：

使用以下原则格式化 Excel 中的数据：

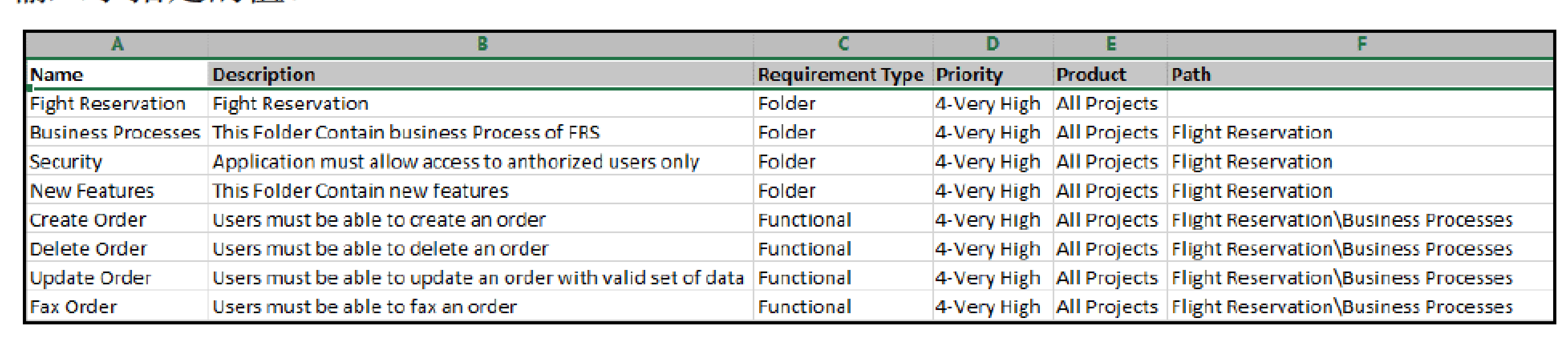
1. 为 ALM 中的需求、测试和缺陷实体分别创建 Excel 工作表。
2. 每个需求、测试或缺陷必须输入单独的行。
3. 每个数据字段必须输入到单独的列。

根据 ALM 工程，每个需求、测试或缺陷都必须有所有要求字段的数据。

用于具有选择列表的 ALM 字段的数据必须准确地与选择列表中输入的数据相匹配。例如，Priority 字段只接受以下的一种输入：5-Urgent，4-Very High，3-High，2-Medium 或 1-Low。

15.2.4　在 Excel 表中格式化需求

格式化 Excel 中的数据，需要使用 ALM 中的数据格式。图 15-4 显示了一个有 9 行的 Excel 工作表。第 1 行提供了不同需求字段的名字。这些字段的名字是 Name、Description、Requirement Type、Priority、Product 和 Path。第 2 行到第 9 行表示具体需求，其中每列分别为各自的字段输入了指定的值。



A	B	C	D	E	F
Name	Description	Requirement Type	Priority	Product	Path
Fight Reservation	Fight Reservation	Folder	4-Very High	All Projects	
Business Processes	This Folder Contain business Process of FRS	Folder	4-Very High	All Projects	Flight Reservation
Security	Application must allow access to anthorized users only	Folder	4-Very High	All Projects	Flight Reservation
New Features	This Folder Contain new features	Folder	4-Very High	All Projects	Flight Reservation
Create Order	Users must be able to create an order	Functional	4-Very High	All Projects	Flight Reservation\Business Processes
Delete Order	Users must be able to delete an order	Functional	4-Very High	All Projects	Flight Reservation\Business Processes
Update Order	Users must be able to update an order with valid set of data	Functional	4-Very High	All Projects	Flight Reservation\Business Processes
Fax Order	Users must be able to fax an order	Functional	4-Very High	All Projects	Flight Reservation\Business Processes

图 15-4　Excel 文档的需求

工作表中的 Path 列不与 ALM 字段相关。使用这列来区分每个需求的层级。例如，Flight Reservation 需求的 Path 列中没有内容，表明本需求是一个父需求。相反情况是，Business Processes 需求的 Path 列中定义的输入表明本需求是 Flight Reservation 需求的一个子需求。Create Order、Delete Order、Update Order 和 Fax Order 需求的 Path 列输入，表明这些需求是 Business Processes 的子需求。

15.2.5　将数据需求从 Excel 导出到 ALM

在 Excel 表中格式化需求后，执行导出流程。

从 Excel 中导出数据的步骤如下：

(1) 打开包含需要导出数据的工作表。将包含要导出数据的行和列选中为高亮状态。

注意：如果工作表的第一行包含标签或字段名称，确定选中部分中没有第一行。

(2) 在 Add-Ins 菜单，选择 Export To ALM 按钮。弹出 HP ALM Export Wizard 对话框。

(3) 输入要导入数据的 ALM 服务器的 URL(如图 15-5 所示)。单击 Next 继续。

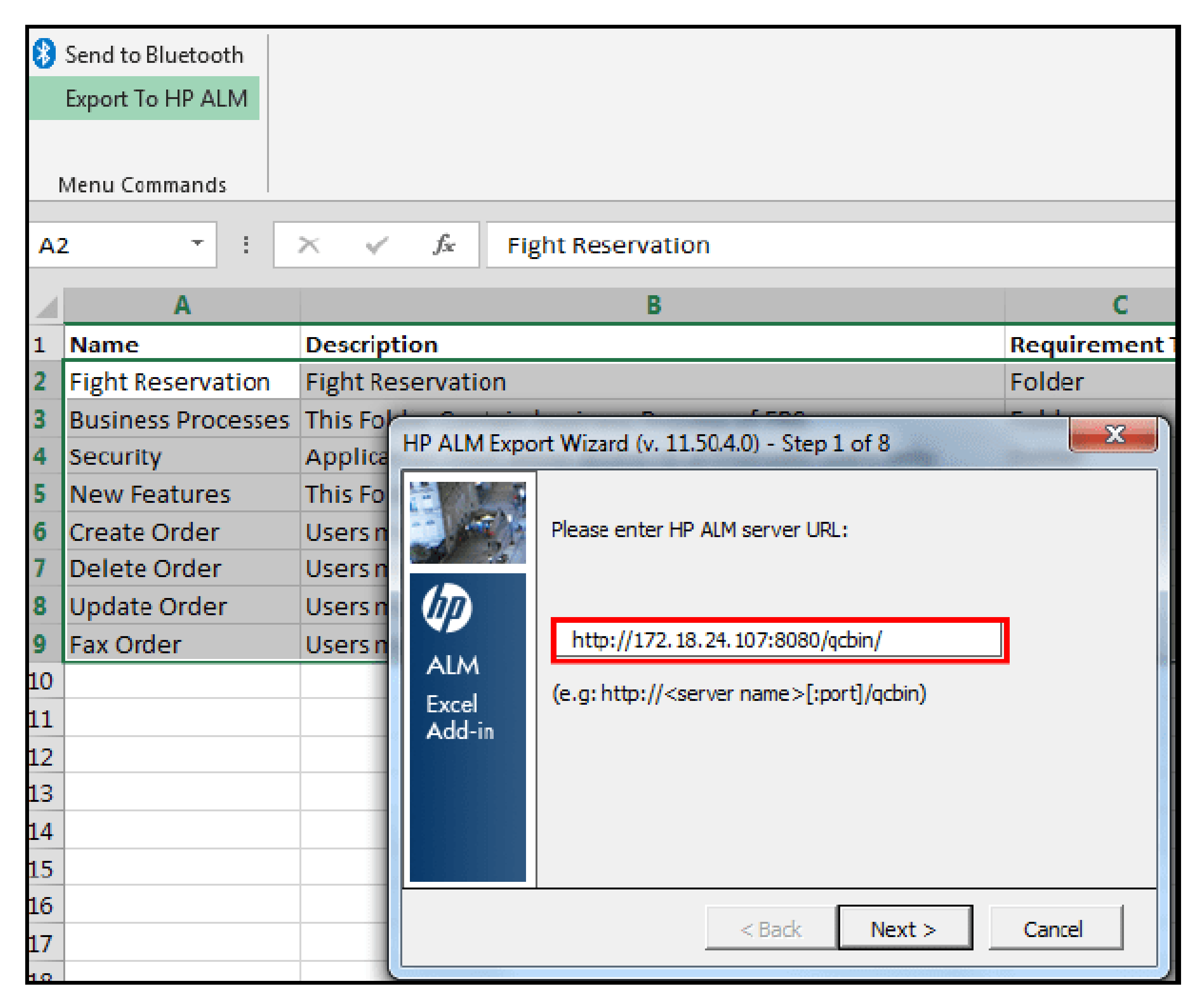


图 15-5　登录质量中心服务器

(4) Excel 需要 ALM 中接收数据工程的用户名和密码。在各自的字段，输入访问 ALM 工程的用户名和密码(如图 15-6 所示)。然后单击 Next 继续。



图 15-6　导出 Wizard

(5) 为 Domain Name 字段选择一个域，为 Project Name 字段选择一个工程。单击 Next 继续。

(6) 选择 Requirements、Tests 或 Defects 代表要导出的数据类型(如图 15-7 所示)。单击 Next 继续。

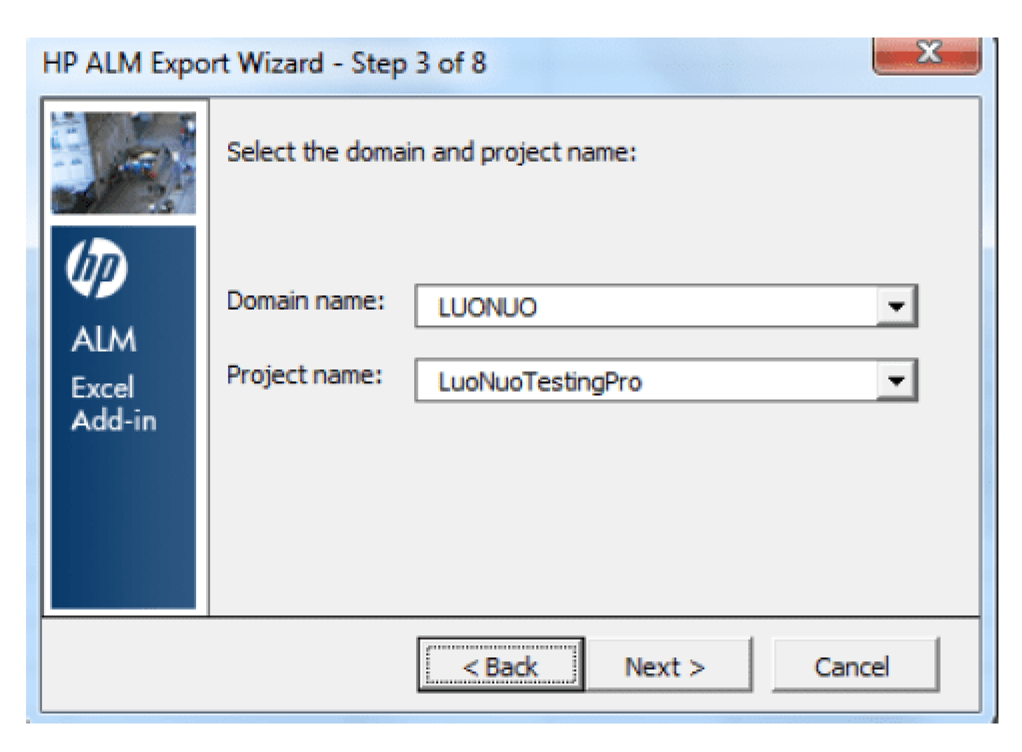


图 15-7　选择项目和数据类型

(7) 选择一个绘图选项来定义如何将工作表中的列与 ALM 中的字段匹配一致。可以从以下选项中选择：

1) Select a Map：可以使用在先前执行中创建的图。在下拉列表中选择。

2) Type a New Map Name：可以创建新图。图名称中列出下次将数据从 Excel 导出到 ALM 的时间，以及后续步骤的定义。

3) Create a Temporary Map：可以创建临时图。图名中不列出下次将数据从 Excel 导出到 ALM 的时间。

4) 单击 Next 继续。

指定导出的需求类型数据(如图 15-8 所示)的步骤如下：

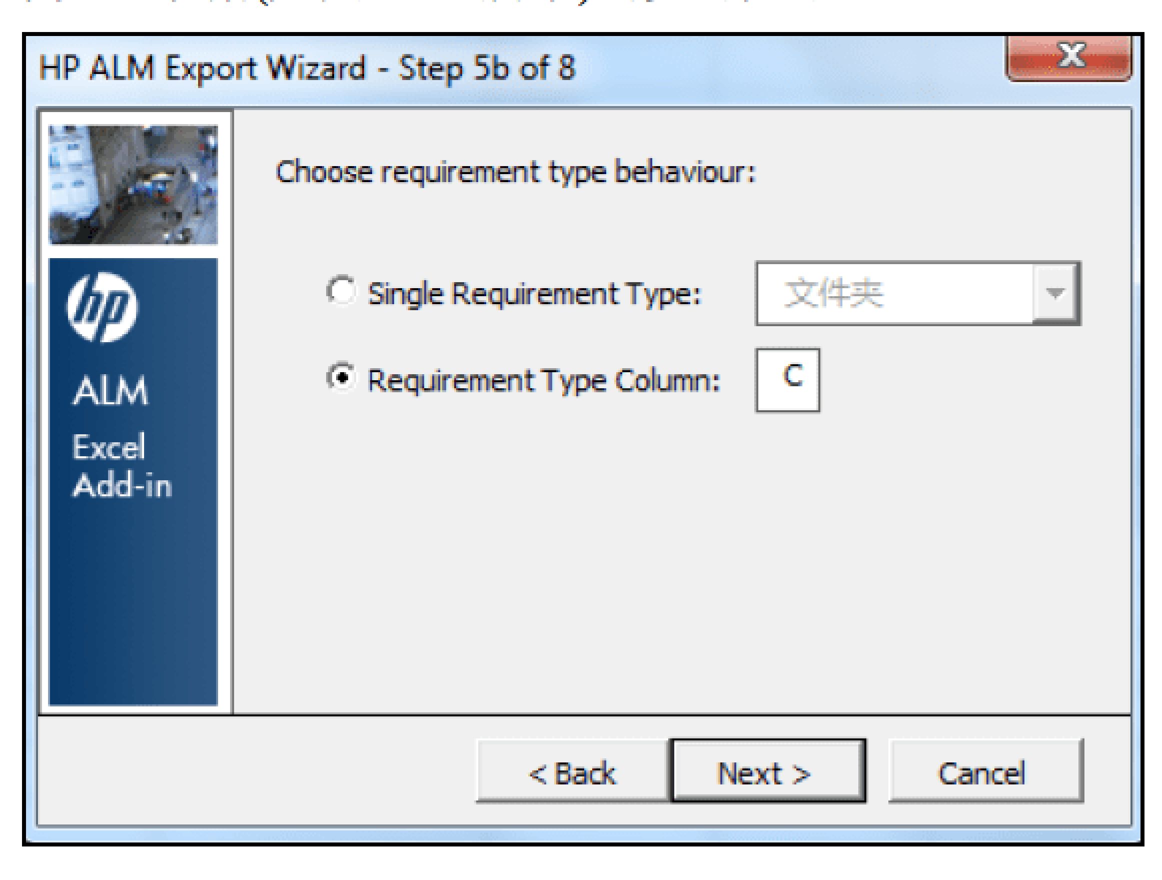


图 15-8　指定需求类型

(1) 如果正在将指定的需求类型数据导出，则选择 Single Requirement Type 选项，然后在右边的字段中指定需求类型。

(2) 如果导出数据是包含带有多种需求的整个需求树，则选择 Requirement Type Column 选项，然后指定包含需求类型数据的 Excel 列字符。

(3) 单击 Next 继续。

向 ALM 映射 Excel 选项(如图 15-9 所示)的步骤如下：

(1) 指定 Excel 工作表中要导出数据到 ALM 中的列。

(2) 在 ALM 字段列表，选择字段，单击显示为向右箭头的图按钮。弹出 Map Field with Column 对话框。

(3) 在 Map Field with Column 对话框，指定符合选择的 ALM 字段的 Excel 列字母。

(4) 单击 OK 将列映射给字段，映射 ALM 中的其他字段，重复第一步和第二步。

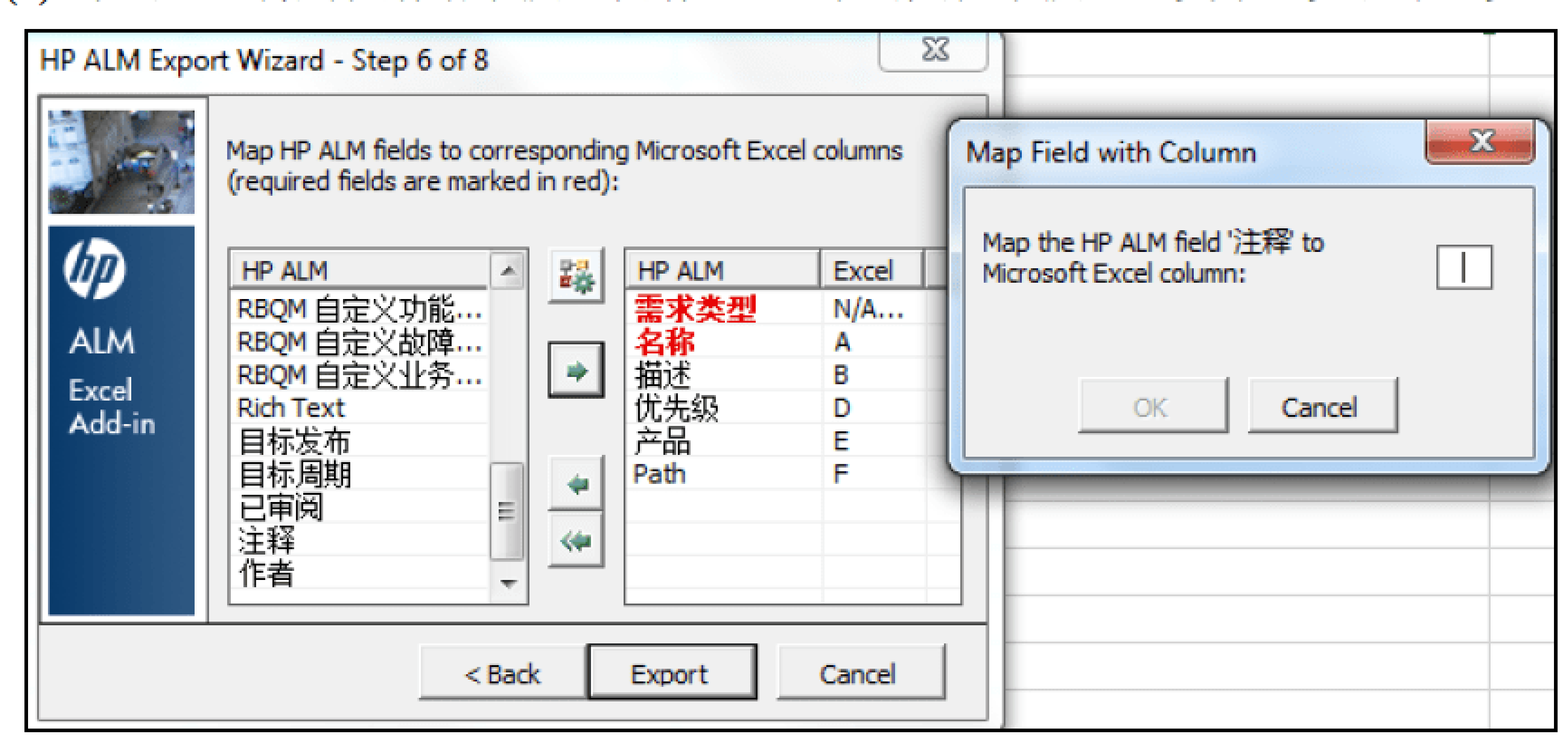


图 15-9　向 ALM 映射 Excel 选项

需求导出(如图 15-10 所示)的步骤如下：

(1) 单击 Export 开始导出过程。

(2) 如果有错误，Export Report Error 将列举在导出过程中遇到的问题。注意这些错误，然后单击 Cancel，在下次运行向导之前修正它们。

(3) 如果没有错误，则消息显示 You have successfully exported the Microsoft Excel worksheet to HP ALM。单击 Finish，关闭向导。

(4) 关闭 Microsoft Excel 返回 ALM，检查导出需求。可能需要单击 Refresh 查看数据。

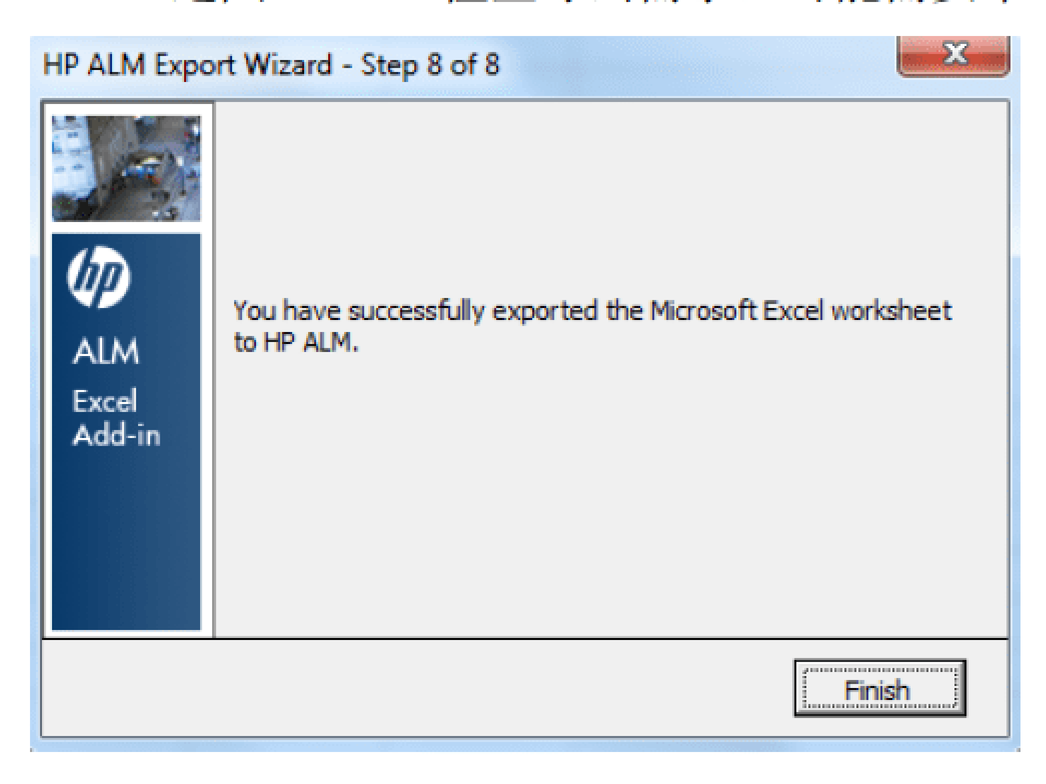


图 15-10　需求导出过程

15.2.6　确认结果

如图 15-11 所示，模块展现出从 Excel 导出的文件夹和需求。

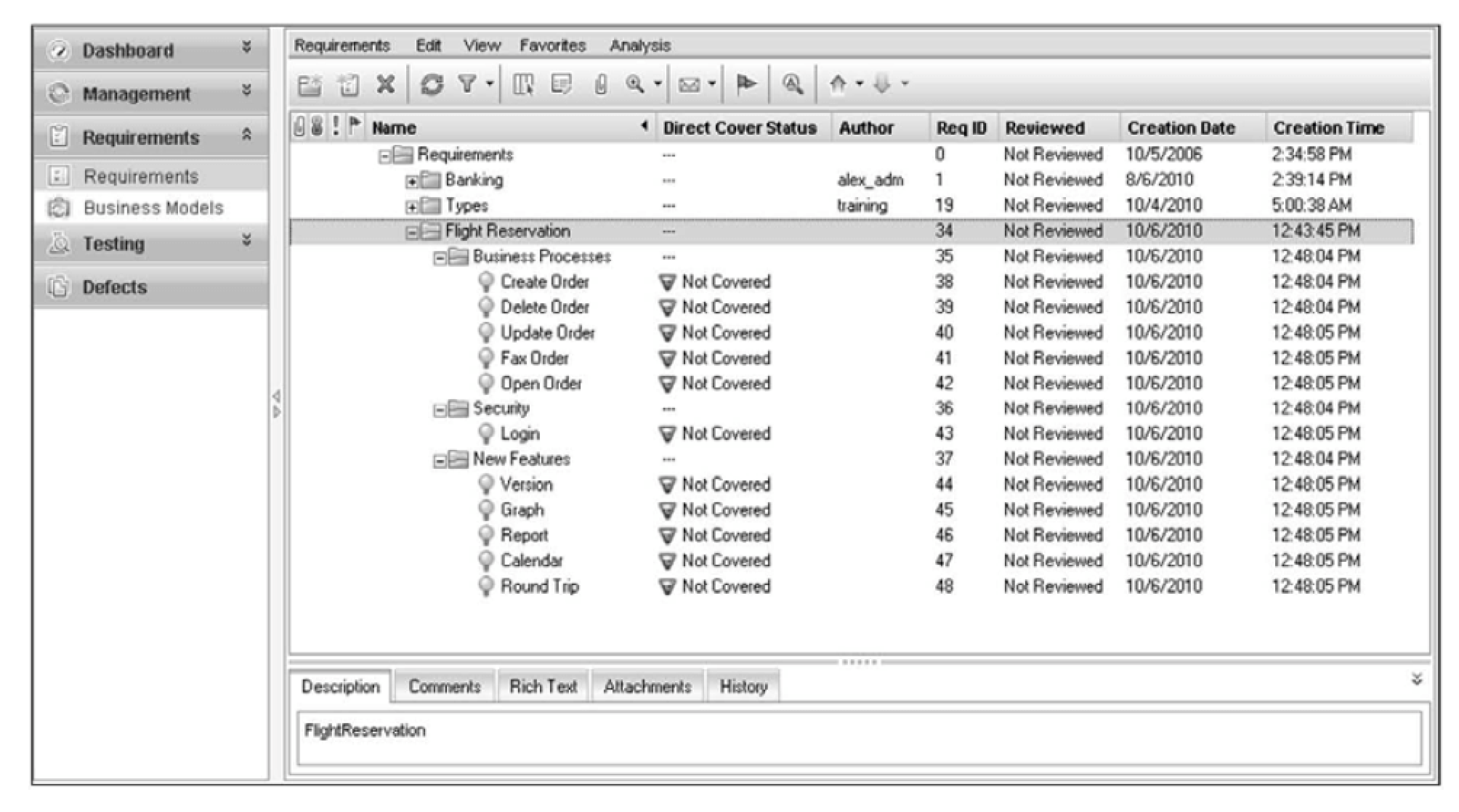


图 15-11　确认结果

15.2.7　在 Excel 中格式化测试

图 15-12 所示的示例中：

	A	B	C	D	E	F	G
1	ID	Test Name	Subject	Description	Step Name	Description(Design Steps)	Expected Result
2	1	2.1.1.1 Ess 用户成功创建请求	SM系统/手工测试/2.1 用户自主服务/2.1.1 提交请求	验证Ess 用户自主创建请求成功	1	在地址栏中输入SM系统的URL，按Enter键或点击Go图标;	SM系统的登录界面打开;
3					2	输入用户名：SelfService.User 密码：SelfService.User 点击"登录"按钮	登录进入SM系统，用户名SelfService.User 在右上角显示
4					3	在左侧导航栏中，点击"提交请求"链接	打开查找请求页面
5					4	检查默认值	各字段的默认值必须正确

图 15-12　格式化测试

1. Test Name、Description、Step Name、Step Description 和 Expected Result 列可以包含用户自定义值。

2. Type 列必须包含与目标 ALM 工程中使用的同类有效值。

3. Subject 列的值必须代表有效的文件夹。

15.2.8　导出测试

图 15-13 所示的是将 Excel 表中的测试导出到 ALM 中：

(1) 打开包含测试步骤的 Excel 表。

(2) 在 Excel 中选中需要导出的测试区域。

(3) 在 Add-Ins 选项卡中，单击 Export To HP ALM 按钮。弹出 ALM Export Wizard-Step 1 of 8 对话框。

(4) 输入服务器名并单击 Next 按钮，ALM Export Wizard-Step 2 of 8 对话框弹出。

(5) 输入用户名和密码并单击 Next 按钮。ALM Export Wizard-Step 3 of 8 对话框弹出。

(6) 输入域名和项目名，单击 Next 按钮。ALM Export Wizard-Step 4 of 8 对话框弹出。

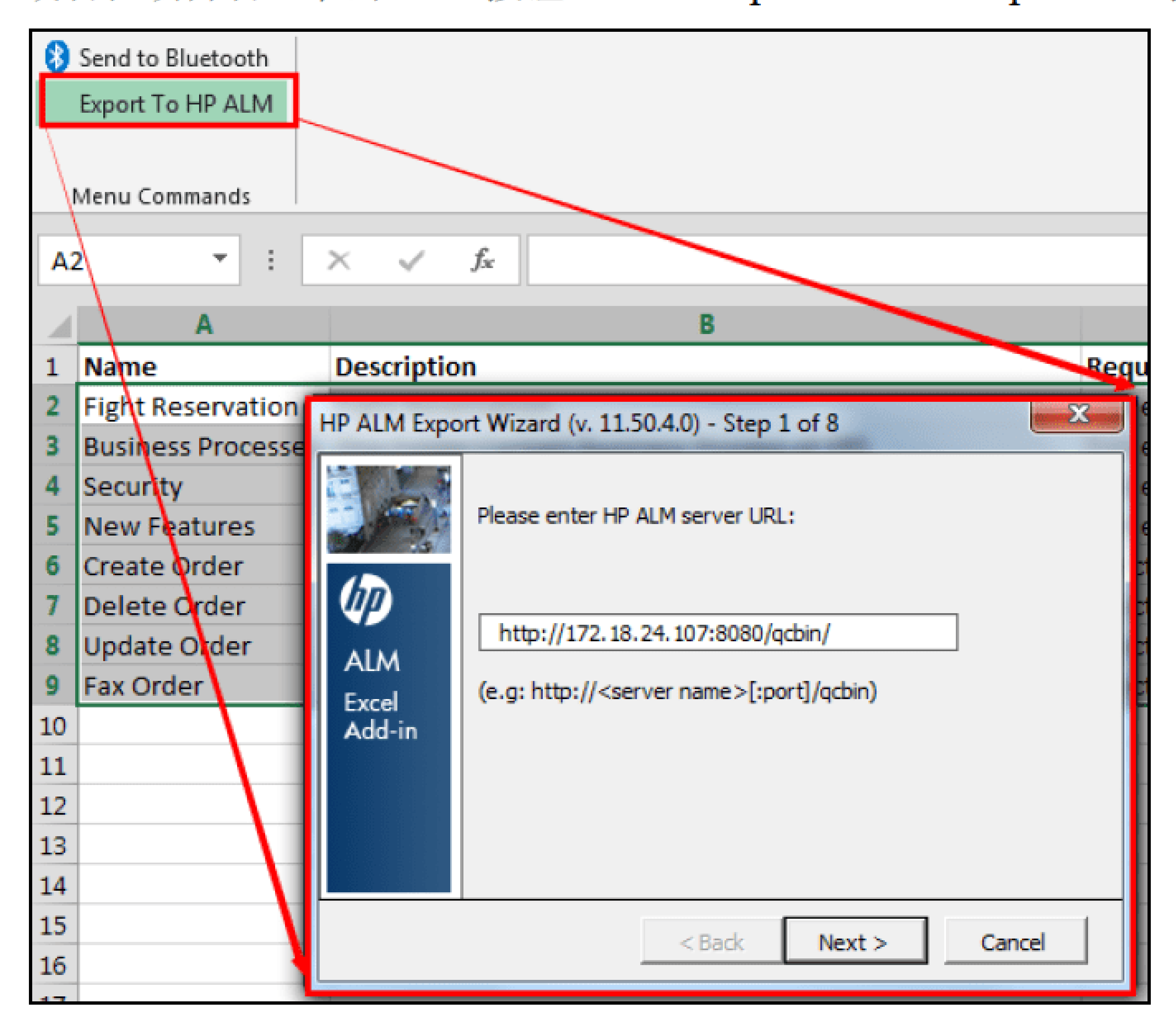


图 15-13　导出测试

(7) 选择需要导出的测试并单击 Next 按钮，ALM Export Wizard-Step 5 of 8 对话框弹出。

(8) 选择 Type a New Map Name，输入一个映射名称并单击 Next 按钮。ALM Export Wizard-Step 6 of 8 对话框弹出。

(9) 单击 Map Fields by Labels 按钮，选择 1 for Map Fields by Row 并单击 Next 按钮。

(10) 单击 Export 按钮。ALM Export Wizard-Step 8 of 8 对话框弹出。

(11) 单击 Finish 按钮。

在 ALM 中验证导出数据的步骤如下：

(1) 使用用户名、密码、域名和项目名称登录到 ALM。

(2) 单击 Test Plan 模块。

(3) 单击 Refresh 按钮刷新所有页面。

(4) 选择 View | Test Plan Tree。

(5) 放大目录，检查导出文件夹和测试。

(6) 单击每个测试，然后在 Design Steps 标签查看设计步骤。

习题与思考题

1. 将下列导出流程中的步骤按正确顺序排列：

1) 选中 Excel 中的数据。

2) 格式化 Excel 中的数据。

3) 检查 ALM 的导入数据，进行必要的添加和修改。

4) 在 Excel 2007 的 Add-Ins 标签，选择 Export to ALM 按钮。

2. 正确或错误：如果 Excel 工作表的第一行包含列名，这些列名必须准确与 ALM 字段名相符。

3. 如果要将 Excel 数据列以下拉列表的形式导出 ALM，哪些细节不能被忽略？

练习：从 Excel 导出需求数据

你的公司正将先前使用 Microsoft Excel 之类的第三方软件进行登录数据测试练习，转换为使用 ALM 测试协调者。你的工作是将 Excel 格式的旧测试数据转换为 ALM 认可的格式，然后将数据从 Excel 导出到 ALM 的项目中。

本练习中，完成以下任务：

第 1 部分：安装 Microsoft Excel 软件

第 2 部分：格式化并选择 Excel 中的数据

第 3 部分：将 Excel 数据导出 ALM

(该练习的指导步骤请参见本章正文)

第16章　报表和分析

1. 应用程序生命周期管理流程图

可以在每个 ALM 模块中生成报表和图表来跟踪和评估项目进程。ALM 的 Requirements、Test Plan、Test Lab 和 Defects 模块提供预定义报表和图表模板。可以使用模板检索需求分析的信息。

惠普应用程序生命周期管理(ALM)通过定义发布和循环来组织和跟踪即将进行的发布。测试流程始于在 Management 模块中定义 Releases。该模块用于根据项目需求、测试和缺陷，累排列业务的优先级别和质量期望。应用程序生命周期管理流程图如图 16-1 所示。

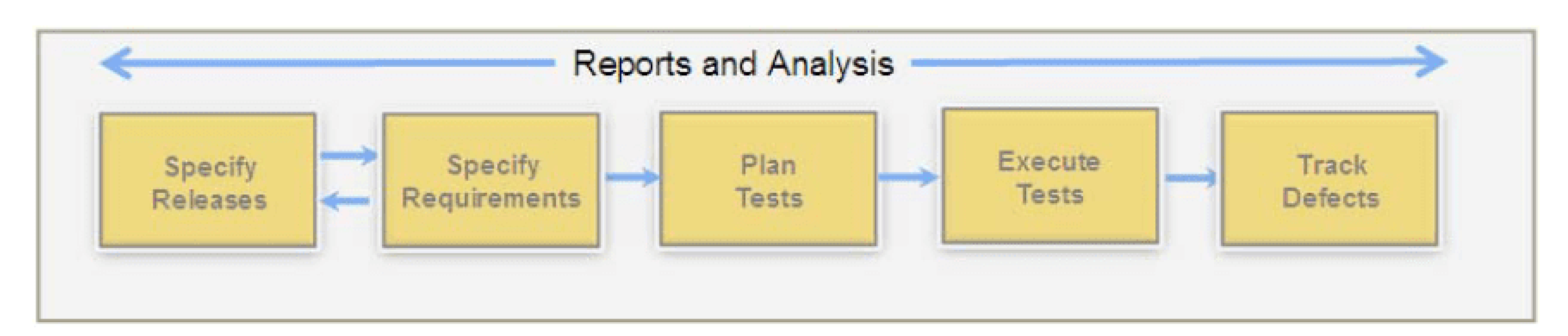


图 16-1　应用程序生命周期管理流程图

2. 报表选项概述

ALM 提供以下选项生成图表和报表：

1) Pre-Defined Reports and Graphs：预定义报表能帮助监控需求和测试覆盖、测试计划、测试运行和缺陷跟踪。我们在 Dashboard 或者在 Requirements 中、Test Plan、Test Lab 或 Defects 模块的工作过程中创建预定义报表。在任一情况下，都可以在 Dashboard 中保存预定义报表以便以后引用。

2) Document Generator：Tools 菜单提供将数据从 ALM 到 Microsoft Word 的 Document Generator 工具。Document Generator 工具帮助将测试数据分类到格式化的 Word 文档。这些测试数据能够指定到测试过程的一个阶段或应用到完整的测试周期。

3) Customer Reports and Graphs：在 ALM 中我们选择创建自己的称为 Customer Reports and Graphs 的报表和图表。这些报表能帮助我们监控需求、测试覆盖、测试计划、测试运行和缺陷跟踪。我们能在 Dashboard 或 Requirements 中、Test Plan、Test Lab 或 Defects 模块的工作中创建客户报表。

16.1　项目报告

1. 预定义报表和图表

使用 Analysis 菜单生成 Reports/Graphs：

1) 导航到要用于报表/图表的 ALM 模块。

2) 从 ALM 菜单栏，选择 Analysis | Reports。子菜单显示模块中可用的报表类型。

3) 选择 Analysis | Reports。子菜单显示模块中可用的图表类型。

选择需要的报表/图表类型。在报表/图表生成任务完成之后，报表/图表输出显示在弹出窗口中。

图 16-2 中显示的 Reports/ Graphs 菜单列举了在 Requirements 模块中可用的标准报表/图表类型。Reports/ Graphs 菜单中也显示了用户保存并喜欢的客户报表/图表类型。

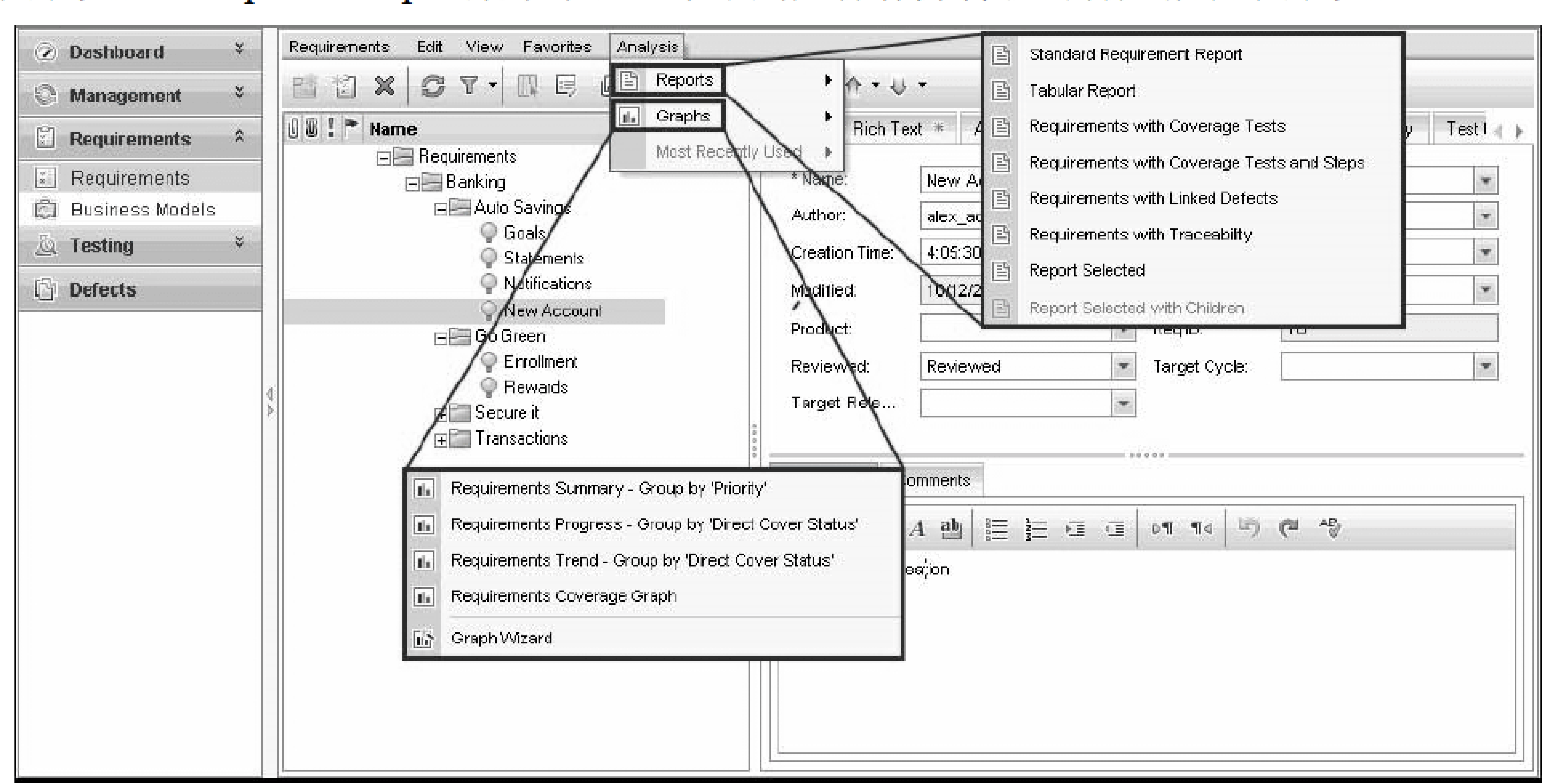


图 16-2　预定义的报告和图表

对于 Analysis | Most Recently Used 中发现的客户报表/图表，以 Public 或 Private 标签为前缀表示对其访问是否有限制。也可以保存为其最喜欢视图的客户报表/图表类型。

Analysis | Report 菜单的 Report Selected 指令生成当前选择记录的报表。例如，可以从需求树中选择多重需求，然后单击 Report Selected 指令生成报表。

2. 分析报告实例

每个 ALM 模块都提供了生成报表和图表的预定义模板。通过默认设置和对其修改这些报表和图表检索需要的信息。可以根据需求组织这些报表和图表。

例如，项目经理可以生成 Defect Report 核实当前关于 New、Open 或 Reopen 状态信息，如图 16-3 中所示。为了创建相似报表，导航到 Defects 模块，并在菜单栏中选择 Analysis | Reports | Standard Defect Report。测试团队可以在状态满足讨论作业和制定行为的过程中使用

这个报表。

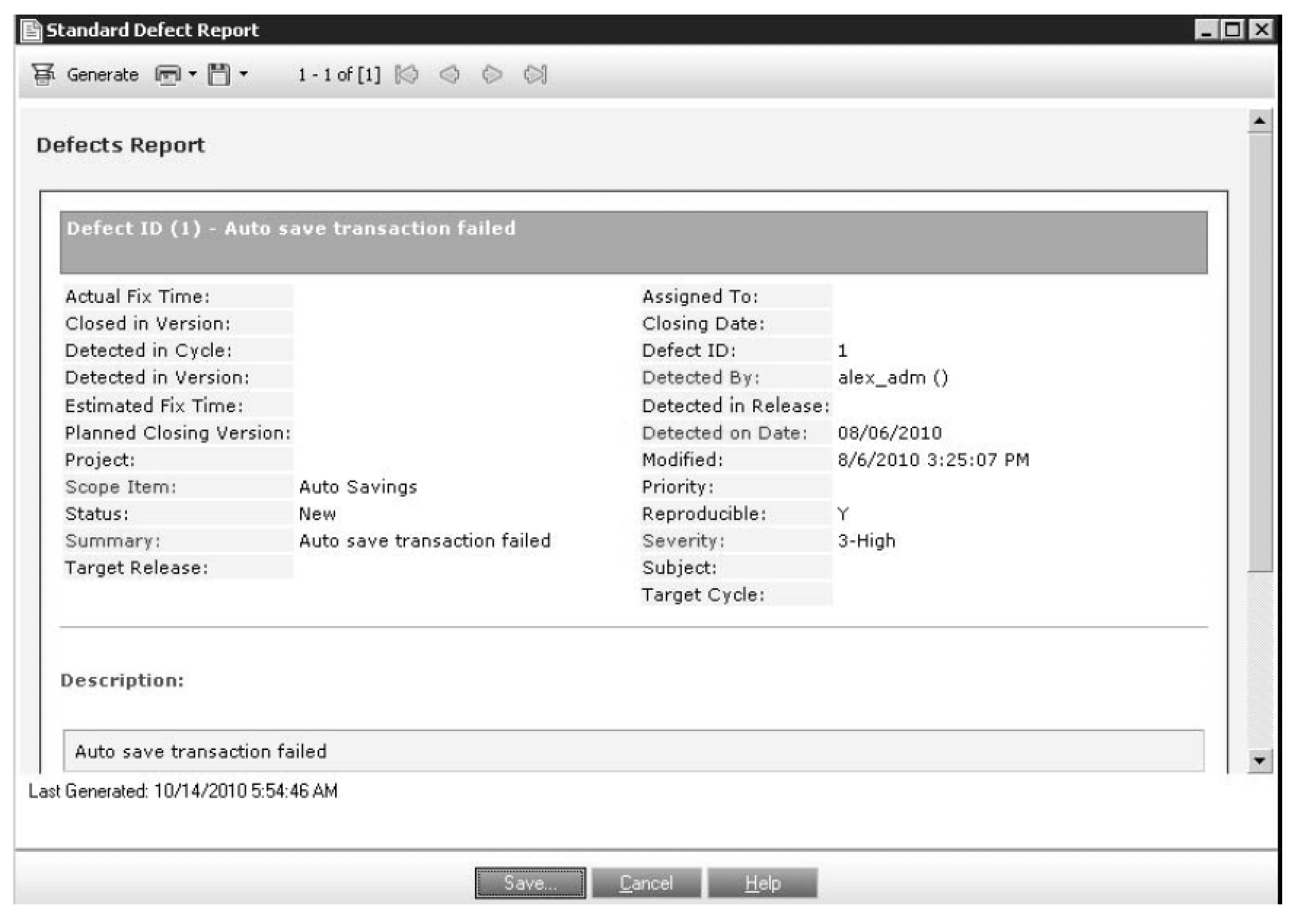


图 16-3　缺陷报告实例

16.1.1　生成项目报告

Analysis View 模块中的 Project Report 报表工具，设计和生成复杂的项目数据报表，如图 16-4 所示。

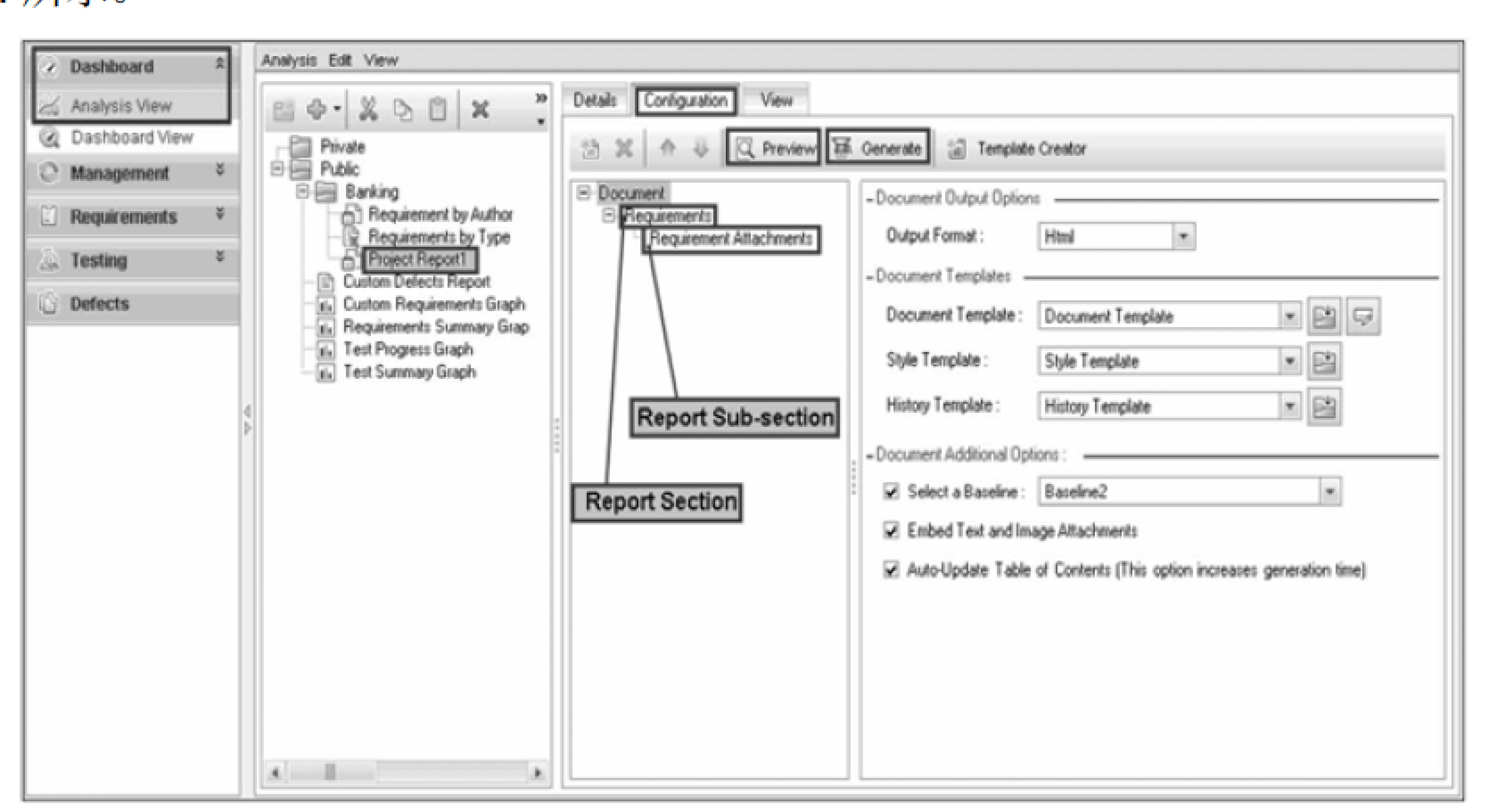


图 16-4　项目报告

借助选择包括在报表部分和定义数据过滤中的实体，使用项目管理员设计的模板为每个实体、用户创建项目报表。

1. 项目报表提供以下优点

1) 使用 MS Word 模板的丰富风格和布局选项。

2) 增强性能。

3) 适用于多种输出格式的单一配置(doc、docx、html)。

4) 集中的模板管理。

2. 创建项目报表

1) 在 ALM 侧边栏的 Dashboard 下选择 Analysis View。

2) 在 Private 或 Public 根文件夹下右击一个文件夹，选择 New Project Report。弹出 New Project Report 对话框。

3) 输入 Project Report Name，单击 OK 按钮。

4) 选择新创建的项目报表，单击 Configuration 标签配置报表。

(1) 选择 Output Format、Document Template、Style Template 和 History Template。

(2) 创建基准报表，选择 Baseline，选择基准线。

(3) 选择 Embed Text And Image Attachments，在报表中嵌入文字和图片附件。

(4) 选择 Auto-Update Table of Contents，ALM 升级报表输出内容实体表格。

1) 右击 Document Root 节点，选择 Add Report Section。选择报表的 ALM 实体并单击 OK 按钮。

2) 添加子部分，右击报表树的一个部分，选择 Add Report Section。选择子部分中的一个 ALM 实体，单击 OK 按钮。

3) 选择一部分或一个子部分进行配置。

(1) 可以有选择地重命名标题选项。

(2) 如果需要，分配一个项目模板。

(3) 若可以，定义数据过滤。

(4) 选择 Keep Hierarchical 分层次地排列报表中的记录。

(5) 单击 Preview 按钮显示报表预览。包含选择格式的文件报表部分的最上边 5 个记录。

(6) 单击 Generate 按钮。保存报表，在 Output Format 字段中打开选择的格式文件 Share Analysis Item 对话框。

对话框能够共享图表查看外部 ALM 客户端。Share Analysis Item 对话框的界面如图 16-5 所示。

3. 启动文档生成器(Document Generator)

以下步骤概述了使用 Document Generator 生成测试数据文档流程的要点：

(1) 启动 Document Generator。

(2) 设置输出文档的内容和格式。

(3) 选择文档中包括的测试数据。

(4) 生成文档。

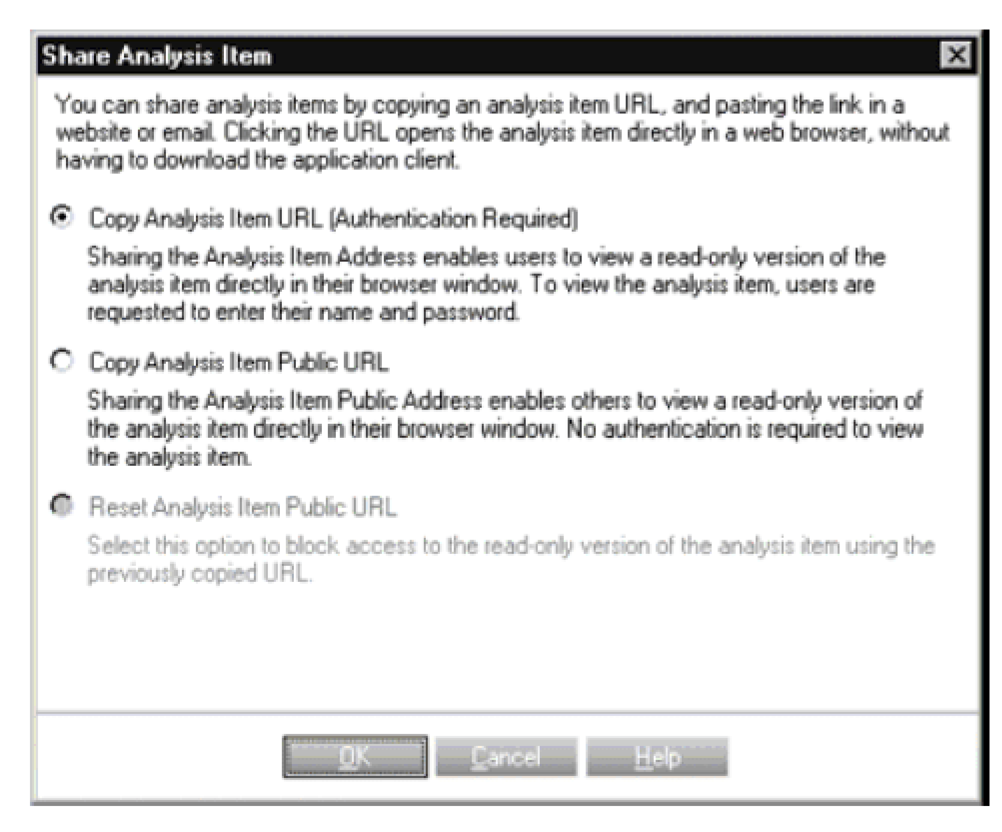


图 16-5　Share Analysis Item 对话框

使用 Document Generator 工具可以生成由需求、测试计划、测试列表、测试执行和缺陷追踪数据组成的复杂项目文档(如图 16-6 所示)。在格式化的 Word 文档中使用 Document Generator 工具分类记录测试数据。在电脑上需要安装 Microsoft Word 2010 或更高版本工具查看 ALM Tools 菜单中的 Document Generator 选项。使用 Document Generator 工具之前，关闭 Generator 选项。使用 Document Generator 工具之前，在格式化的 Word 文档中使用 Document Generator 工具分类记录测试数据。在电脑上需要安装 Microsoft Word 2010 或更高版本工具查看 ALM Tools 菜单中的 Document Generator 选项。使用 Document Generator 工具之前，关闭电脑上的 Microsoft Word 程序。

Analysis 菜单的 Report Selected 命令生成目前选择记录的报表。例如，可以从需求树选择多重需求，并单击 Report Selected 命令生成报表。

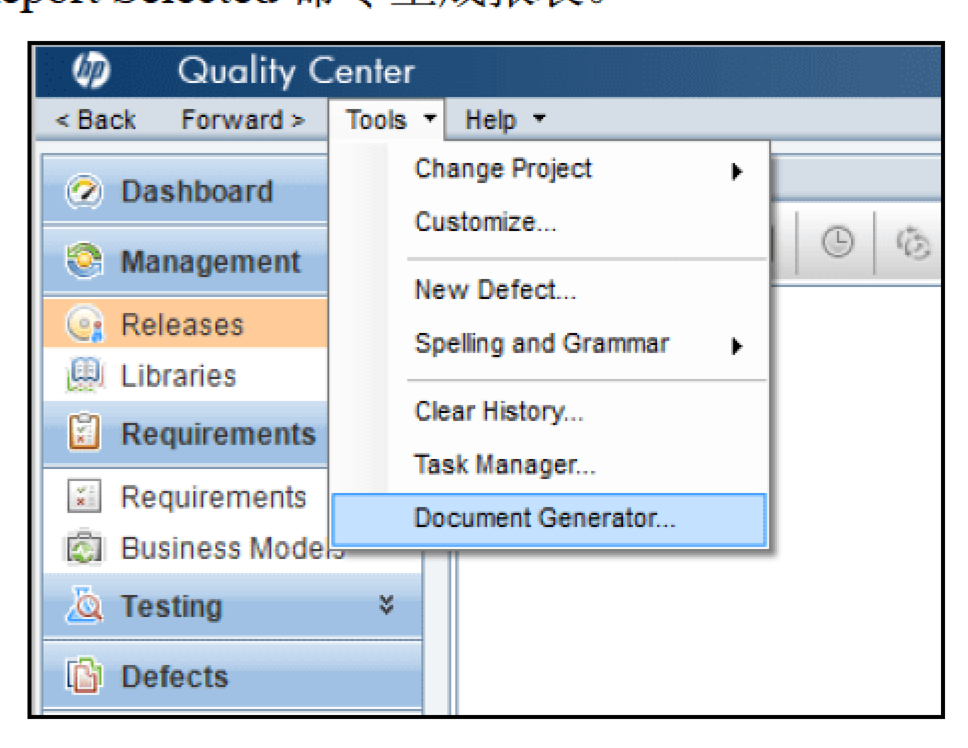


图 16-6　文档生成器

设置文档选项(Document Options)，使用 Document Generator 窗口定义输出文档的格式和布局。定义输出文档的总体格式和布局的步骤如下：

(1) 在 Document Generator 窗口的 Document Generator 树中，单击 Document 节点。Document Generator 窗口中有四个标签：Document Settings、Options、Logo 和 Customization。

(2) 单击 Document Settings 标签，在文档标题页输入 Title 字段、Author 部分和 Description 字段的详细内容(如图 16-7 所示)。

(3) 单击 Options 标签，选择想要的页面布局选项。例如，在文档中添加内容标签和索引。

(4) 在文档中放入公司标签，步骤如下：

① 单击 Logo 标签。

② 在本地驱动中，检索选中的图片文件，单击 Load From File。

(5) 单击 Customization 标签，单击 Page Setup 按钮设置页面属性，例如边缘、方向和页面尺寸。

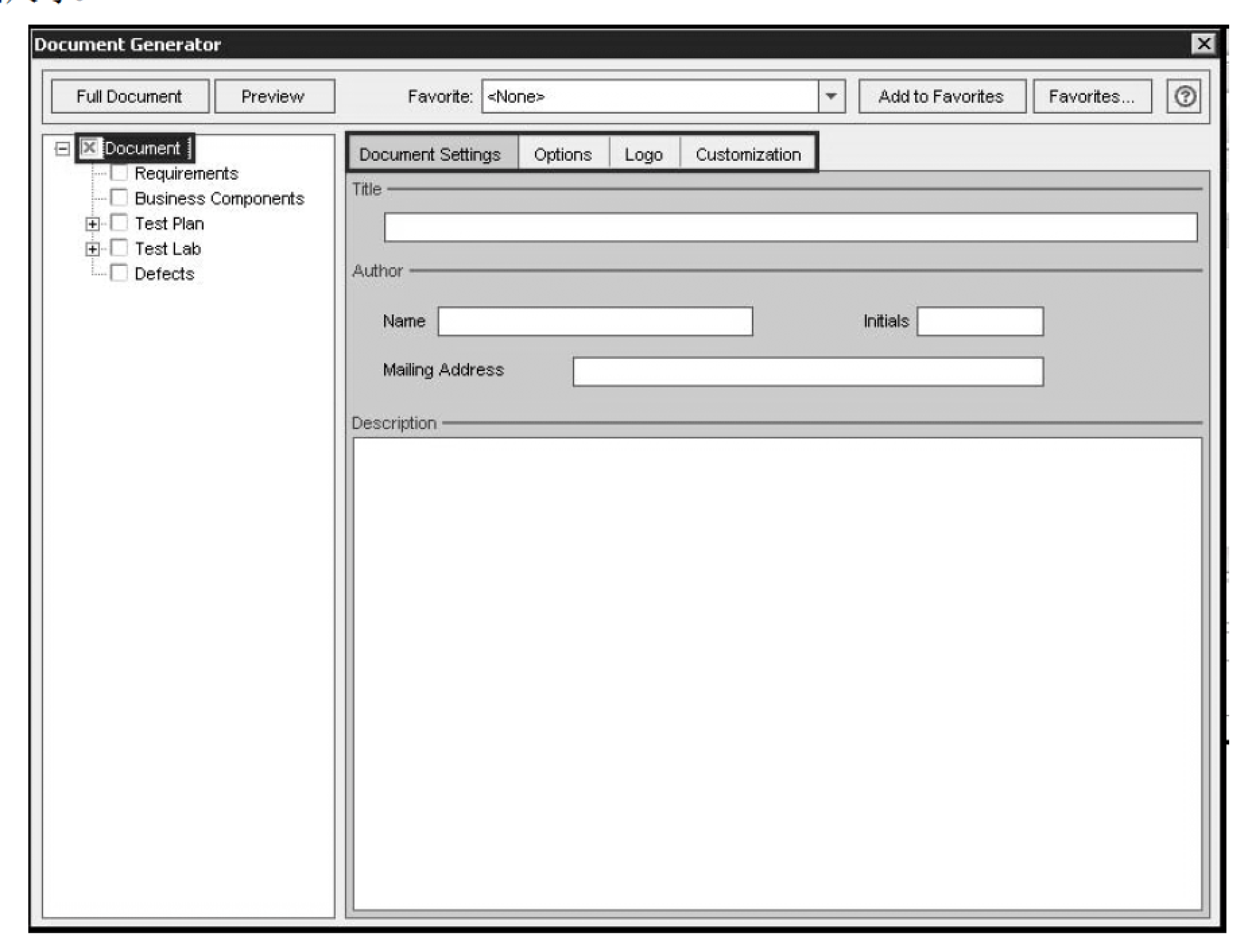


图 16-7　文档设置

16.1.2　生成文档

定义了输出文档的格式和布局之后，需要指定输出文档中需要包含的数据。

指定输出文档中需要包含的数据的步骤如下：

(1) 在 Document Generator 树，勾选要包括到文档中相关的测试数据单选框。单击单选框旁的节点显示所选的指定测试数据选项。

(2) 设置选中测试数据的需要选项。例如，对于 Test Plan 文档，可以为 Design Steps 过滤 Tests 并选择 Layout 和 Properties。图 16-8 显示了你能够选择的不同选项。

(3) 向文档中添加更多数据，重复步骤(1)和(2)。

注意：Document Generator 树状结构中选择的数据在同一文档中生成独立章节。

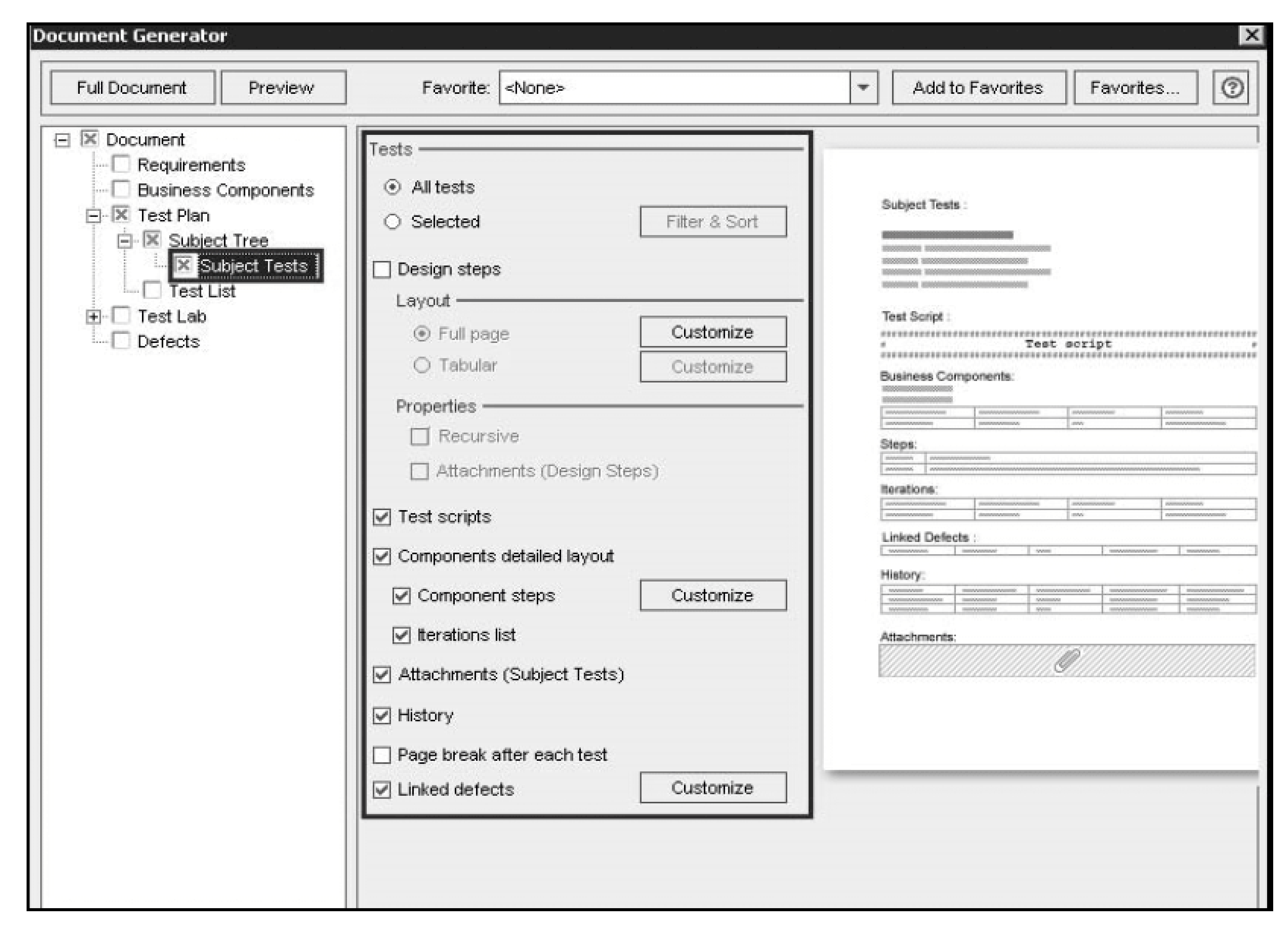


图 16-8　选择数据

定义输出文档格式和布局之后，指定包含在输出文档中指定的数据。如图 16-9 所示，生成文档功能的界面。操作步骤如下：

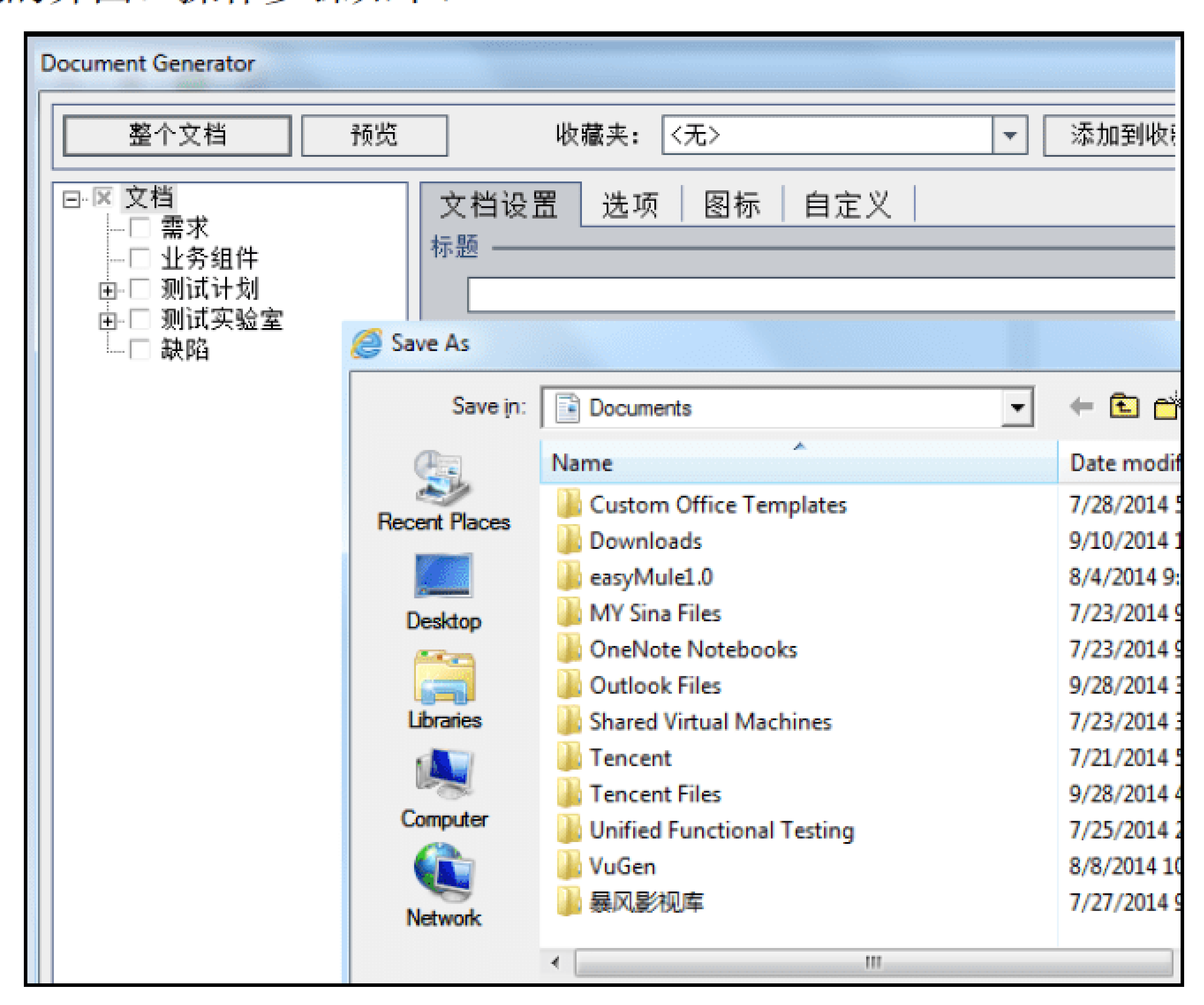


图 16-9　生成文档

指定需要包括在文档中的数据：

(1) 在 Document Generator 树，勾选要包括到文档中的相关测试数据单选框。单击单选框旁的节点显示所选的指定测试数据选项。

(2) 设置选中测试数据的需要选项。例如，对于 Test Plan 文档，可以为 Design Steps 过

滤 Tests 和选中 Layout 和 Properties。图 16-9 显示了可能选择的不同选项。

(3) 为文档添加更多数据，重复步骤(1)和(2)。

注意：Document Generator 树中选择的数据在同一文档中生成独立章节。

16.2　定制报告和图表

如图 16-10 所示，我们可以定义图表、标准报表、客户报表、Excel 报表和使用 Dashboard 面板页的分析条目。在侧边栏，单击 Dashboard，打开 Dashboard。

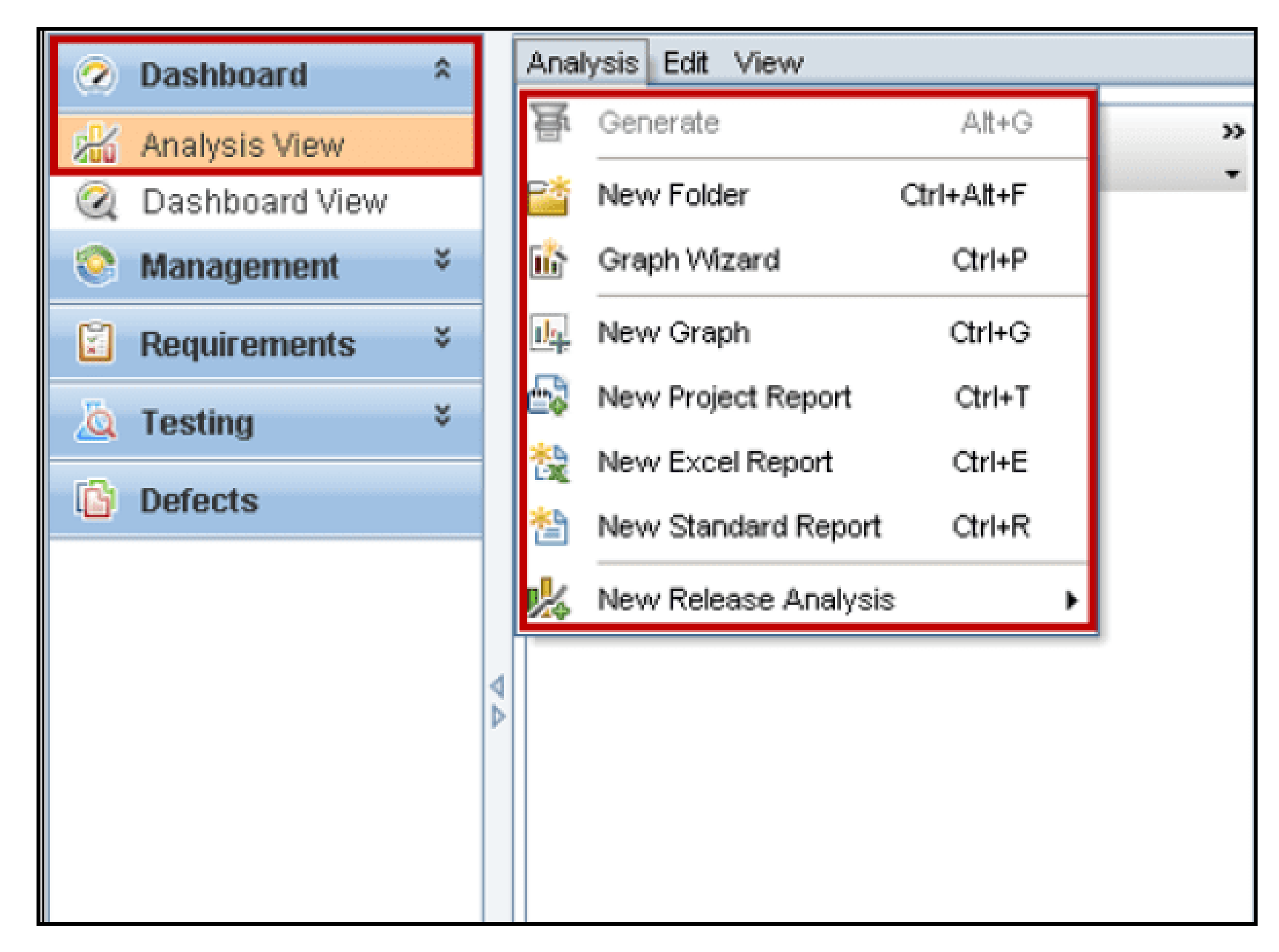


图 16-10　定制报告和图表

Dashboard 包含以下模块：

1. Analysis View 模块：用来创建和管理图表、标准报表和 Excel 报表。

2. Dashboard View 模块：用来创建和管理在单独页面上显示多重图表的面板页。

在 ALM 模块保存报表/图表的步骤如下：

(1) 生成报表并定义所有需要的客户设置。

(2) 当有显示的报表输出，单击 Save，出现 Save As 对话框。

(3) 在 Name 字段，输入你希望的名字。

(4) 选择 Public 或 Private。Private 在私人文件夹中存储最喜欢的视图并通过评估报表输出限制其他用户。Public 在通用文件夹中存储最喜欢的视图使所有指定项目的用户能够访问报表输出。

(5) 单击 New Folder 按钮，生成 Public 或 Private 文件夹。输入文件夹名字，单击 OK 按钮。

(6) 单击 Save，向文件夹添加报表。

注意：选择 Analysis | Most Recently Used 查看添加到文件夹的报表。

新的标准报表，在 Dashboard 中创建标准报表显示 Requirements、Test Plan、Test Lab 和

Defects 模块数据。创建新的标准报告的界面如图 16-11 所示。

图 16-11　新标准报告

注意：也可以从 Requirements、Test Plan、Test Lab 和 Defects 模块创建标准报表。

(1) 在 Dashboard 中单击 Analysis View 模块。

(2) 在分析树中，选择要添加报表的文件夹。

(3) 单击 New Item 按钮，选择 New Standard Report。弹出 New Standard Report 对话框。

(4) 在 Entity 下选择创建报表的数据设置。

(5) 在 Standard Report Type，选择需要创建的报表类型。

(6) 在 Standard Report Name，输入报表名字。

(7) 单击 OK，报表添加到分析树。

(8) 单击 Details 标签，Details 标签显示如表 16-1 所示的字段。

表 16-1　Details Tab 支持的字段和描述

字　段	描　述
Entity	表示报表示例的数据设置
Type	表示分析条目类型
Sub Type	表示报表的类型
Name	表示报表的名字
Last Modified	表示上次修改报表的日期和时间
Modified By	表示上次修改报表的用户
Owner	表示创建报表的用户。仅仅针对所有者修正公开报表权限限制
Description	表示报表的描述

16.2.1　可用的报告类型

在 Dashboard 中创建标准报表(如表 16-2 所示)，选择一种报表类型。不同的报表类型适

用于不同数据实体。

表 16-2　报表和描述

报　表	描　述
Components Report	Business Components 模块的组件列表
Subject Components Report	Business Components 模块的分层组件列表
Requirement Report	Requirements 模块的需求列表
Business Process Tests	Business Processes 测试报表
Subject Test Report	Test Plan 模块的测试子层列表
Test Plan Report	Test Plan 模块的测试列表
Cross Test Set Report	Test Lab 模块的测试设置列表
Execution Report	列举具有测试状态的测试实例
Test Set Hierarchy Report	列举 Test Lab 模块的测试设置文件夹层级
Defect Report	列举 Defects 模块缺陷
Runs	列举 Test Lab 模块的 Test Runs

16.2.2　定制标准报告

可以定义 ALM 报表的外观和内容。分别设置主报表和每个子报表的配置(如表 16-3 所示)。配置标准报表的步骤如下：

(1) 在分析树中，选择一个报表。

(2) 单击 Configuration 标签，在 Page 下，设置在每页显示的主报表的条目数量，限制每页的条目数量，选择 Limit Items Per Page To 并制定每页的条目数量。

(3) 显示页面中的所有条目，选择 All Items in One Page。

(4) 在 Template 中，使用 ALM 默认报表模板或自己的模板。

(5) 添加子报表，单击 Add Sub-Report 按钮。

(6) 在标签左边，选择主报表或想要配置的子报表。

(7) 在 Filter，定义或清理过滤和分类属性。

(8) 单击 Set Filter/Sort 按钮，根据选择的标准过滤和排列数据。

(9) 单击 Clear Filter 按钮清理所有过滤和排列属性。

(10) 在 Fields 中，设置报表的字段和它们的顺序。

(11) 选择 All Fields(自动布局)显示报表中的所有字段。

(12) 选择 Custom Fields(布局)，单击 Select Fields 按钮以选择字段设置顺序。

(13) 在 Options，根据报表类型选择以下选项。

(14) 单击 View 标签生成升级设置的报表。

表 16-3　设置选项

选　项	描　述
附件	显示相关的附件列表
Grid View	用 Grid 显示报表
History	显示需求、测试、缺陷等所有改变的列表
Keep Parent-Child Order	显示具有子需求的需求主题。选择这个选项使定义的过滤和排列属性无效
Rich Text	显示具有子需求的需求主题。选择这个选项使定义的过滤和排列属性无效
Show Full Coverage	显示树中每个需求的测试覆盖
Show Paragraph Number	显示树中每个需求的分配等级数。注意分配给每个需求的与独立 Req ID 无关的数字
Show Steps only for “Failed” Runs	如果添加 Run Steps 子报表，这些步骤只列举 Failed 运行

图 16-12 中显示设置缺陷报告的界面。

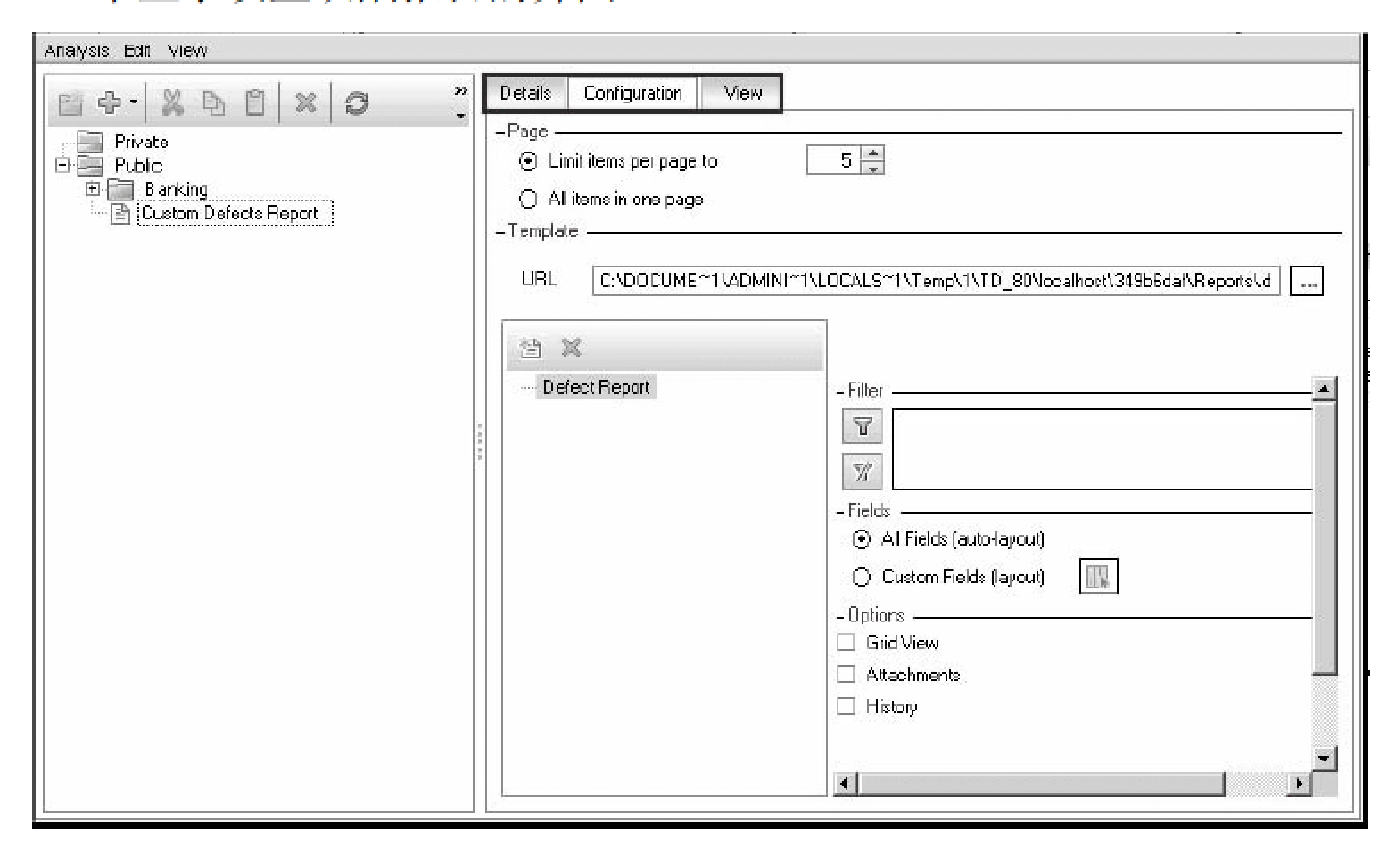


图 16-12　定制缺陷报告

16.2.3　添加子报表

创建报表后，可添加子报表。子报表添加到与父报表相关的另外信息层。例如，创建缺陷报表，可以添加连接需求的子报表。报表之后就显示连接到每个缺陷的需求。对每个子报表，可以添加更深层次的子报表。每一层都可以包含多层子报表。

1. 添加子报表(如图 16-13 所示)。

1) 在分析树，选择一个报表，单击 Configuration 标签。

2) 在 Report 面板，选择要进一步添加子报表的报表或子报表。

3) 单击 Add Sub-Report 按钮。在 Type 列表，选择一个子报表，单击 OK。或者右击报表，从 Add Sub-Report 列表中选择一个子报表。子报表添加到 Report 列表中(如表 16-4 所示)。

2. 删除子报表，选择子报表，单击 Delete Sub-Report 按钮。或者右击报表，选择 Delete Sub Report。如果删除一个父报表，其所有的子报表也会被删掉。

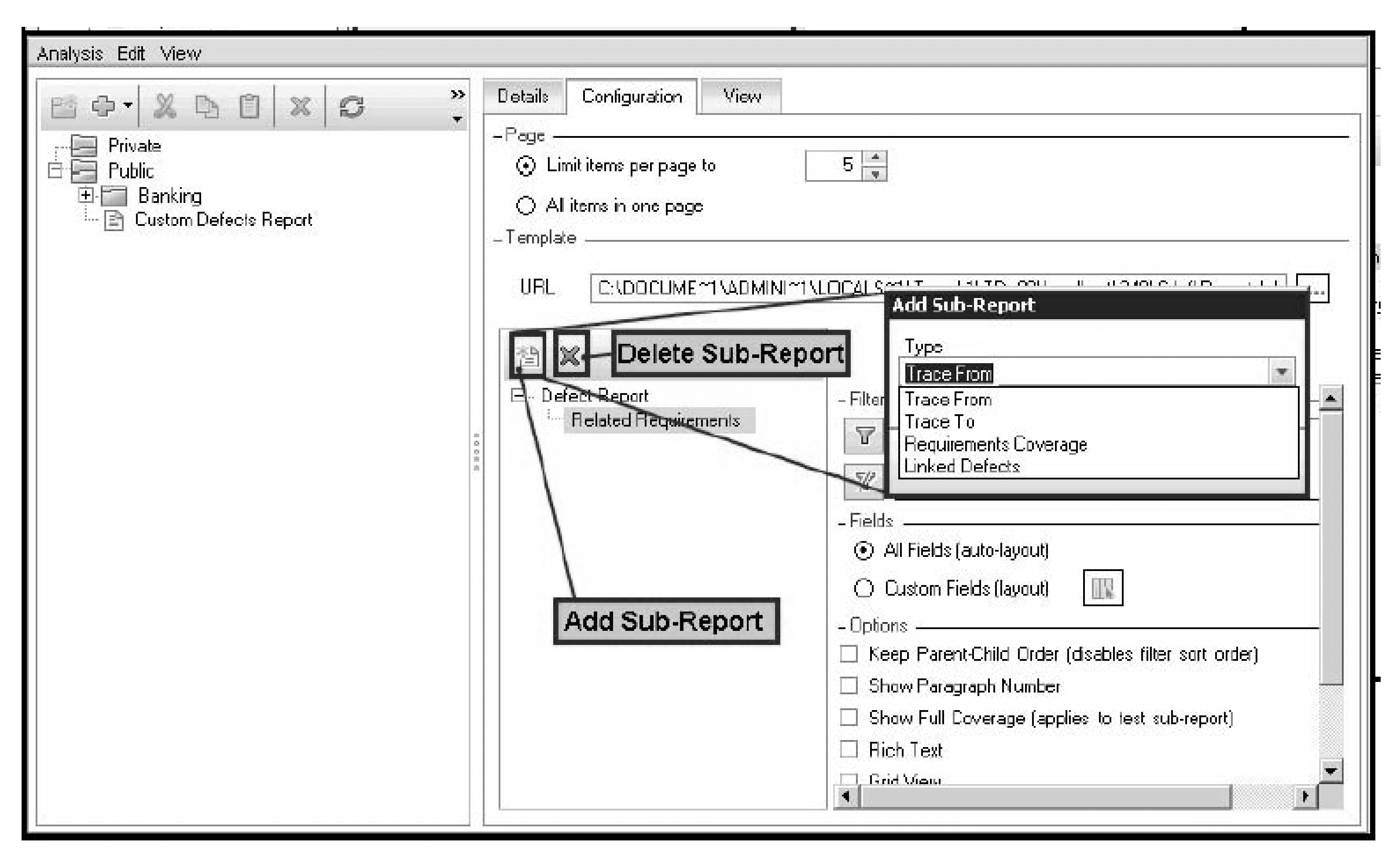


图 16-13　添加子报表

表 16-4　子报告列表

报　　表	描　　述
Contained Test Sets	在测试集文件夹中列举测试集
Contained Tests	列举测试集的测试实例
Coverage Requirements	列举一个测试覆盖的需求信息
Design Steps	列举测试的设计步骤
Linked Defects	列举连接到记录的缺陷
Linked Entities	列举连接到缺陷的所有实体
Parent Test	列举测试运行的父测试实例
Related Requirements	列举连接到缺陷的需求
Requirements Coverage	列举覆盖需求的测试
Run Steps	列举测试运行的运行步骤
Runs	列举测试实例的所有运行
Source Execution Test	列举连接到缺陷的执行测试实例
Source Run	列举连接到缺陷的测试运行
Source Test	列举连接到缺陷的测试
Trace To	列举跟踪到需求的需求
Trace From	列举跟踪自需求的需求

3. ALM 图表类型

使用 ALM 图分析工作进程和测试流程中累计的数据关系。ALM 中以下的图表类型是可

用的：

1) Summary Graph：每个 ALM 模块提供其支持的指定任务的总结图。图中显示测试流程中定义的需求、测试、测试集内测试或缺陷的总量。

2) Progress Graph：每个 ALM 模块提供针对模块支持任务的进程图。这类图显示特定时期内需求、测试、测试集内测试或缺陷的累积。

3) Trend Graph：Requirements、Components、Test Plan 和 Defects 模块提供支持的特定任务的趋势图。这类图显示特定时期指定字段的变化历史。

4) Age Graph：这类图针对指定的 Defects 模块。其概述了所有报表缺陷的周期。缺陷生命周期从创建报表开始，在关闭时结束。

4. 图表向导

图表向导如图 16-14 所示。

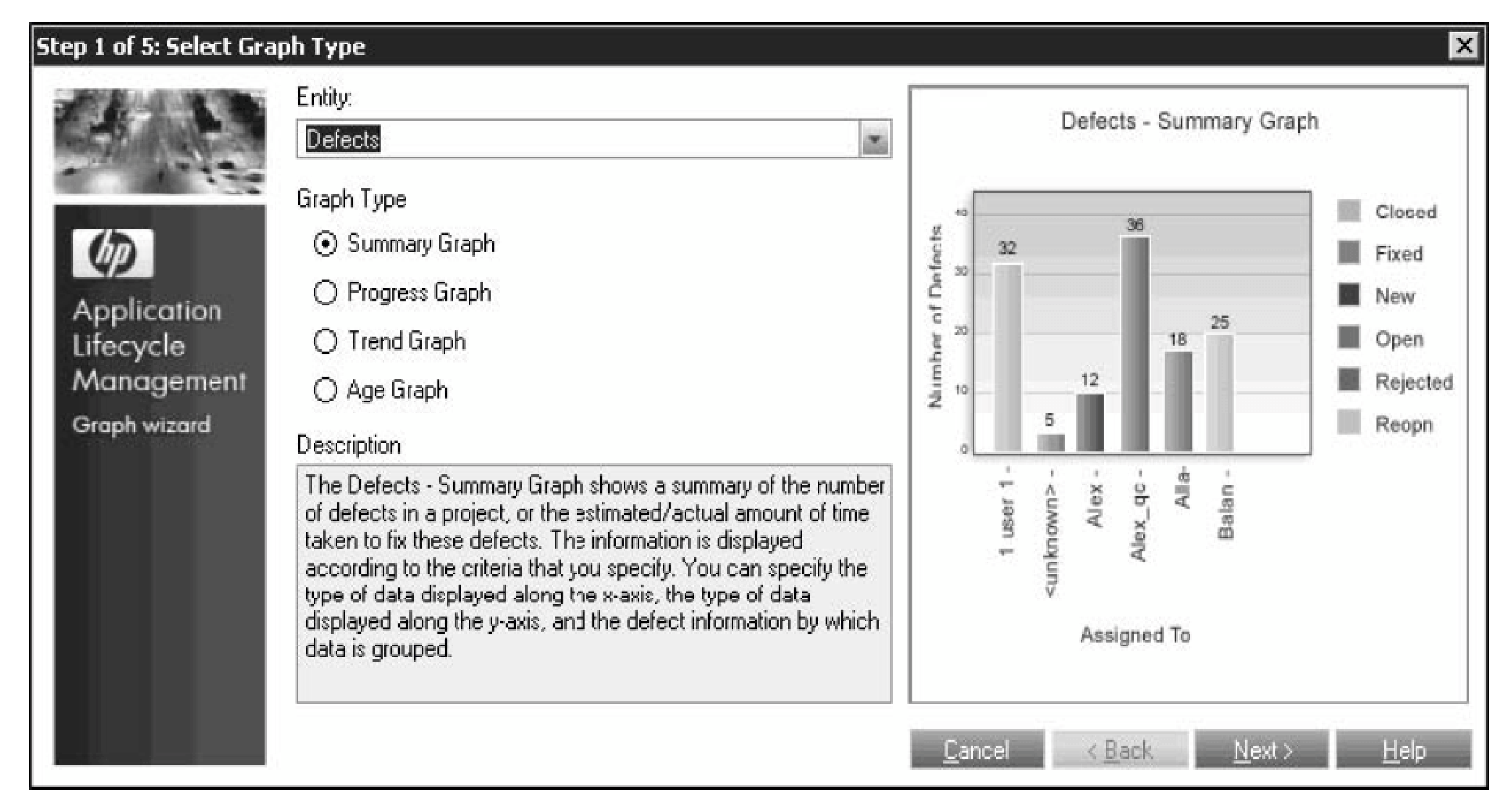


图 16-14　图表向导

使用 Graph Wizard 生成新图。向导带你完成生成新图表的步骤。运行 Graph Wizard：

1) 在菜单栏，选择 Analysis | Graph Wizard。弹出 Graph Wizard 对话框。

2) 选择 Entity 和图示类型并单击 Next。

3) 选择 Project Selection 并单击 Next。

4) 选择过滤选项并单击 Next。

5) 根据图中分组的数据和 X-Axis Field 数据选择字段，单击 Next。

6) 选择 Graph Name 和 Destination Folder，单击 Finish 确认设置生成图表。图在图示窗口中显示出来。

5. 配置标签–图表

Tab-Graphs 是配置标签，如图 16-15 所示。这个标签使选择数据包含到图表和设置图显示选项中。可以在图示数据中包含多重项目。

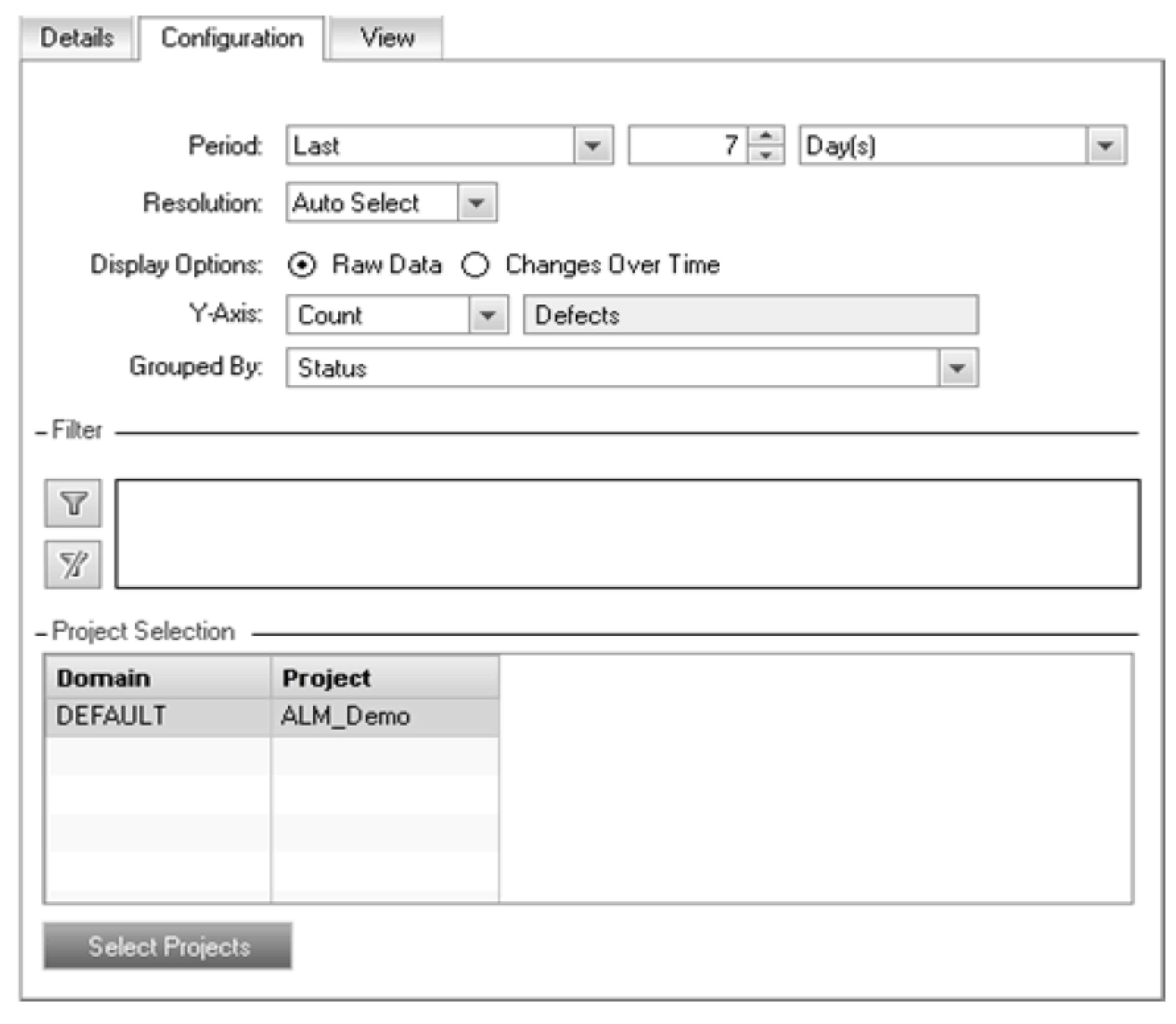


图 16-15　配置标签

16.2.4　不使用向导创建图表

1. 在 Dashboard 中创建图表

1) 在 Dashboard 中，单击 Analysis View 模块。

2) 在分析树中，选择要添加图表的文件夹。

3) 单击 New Item 按钮，选择 New Graph。弹出 New Graph 对话框。

4) 在 Entity 下，选择要创建图表的模块。

5) 在 Graph Type，选择要创建的图表的类型。

6) 在 Graph Name 中输入图表名称。

7) 单击 OK，将图表添加到分析树。

8) 单击 Details 标签。

9) 可以在 Configuration 标签中配置图表内容。查看 View 标签的图表。

图 16-16 中说明上述步骤的选项。

表 16-5 解释了英文字段和中文描述。

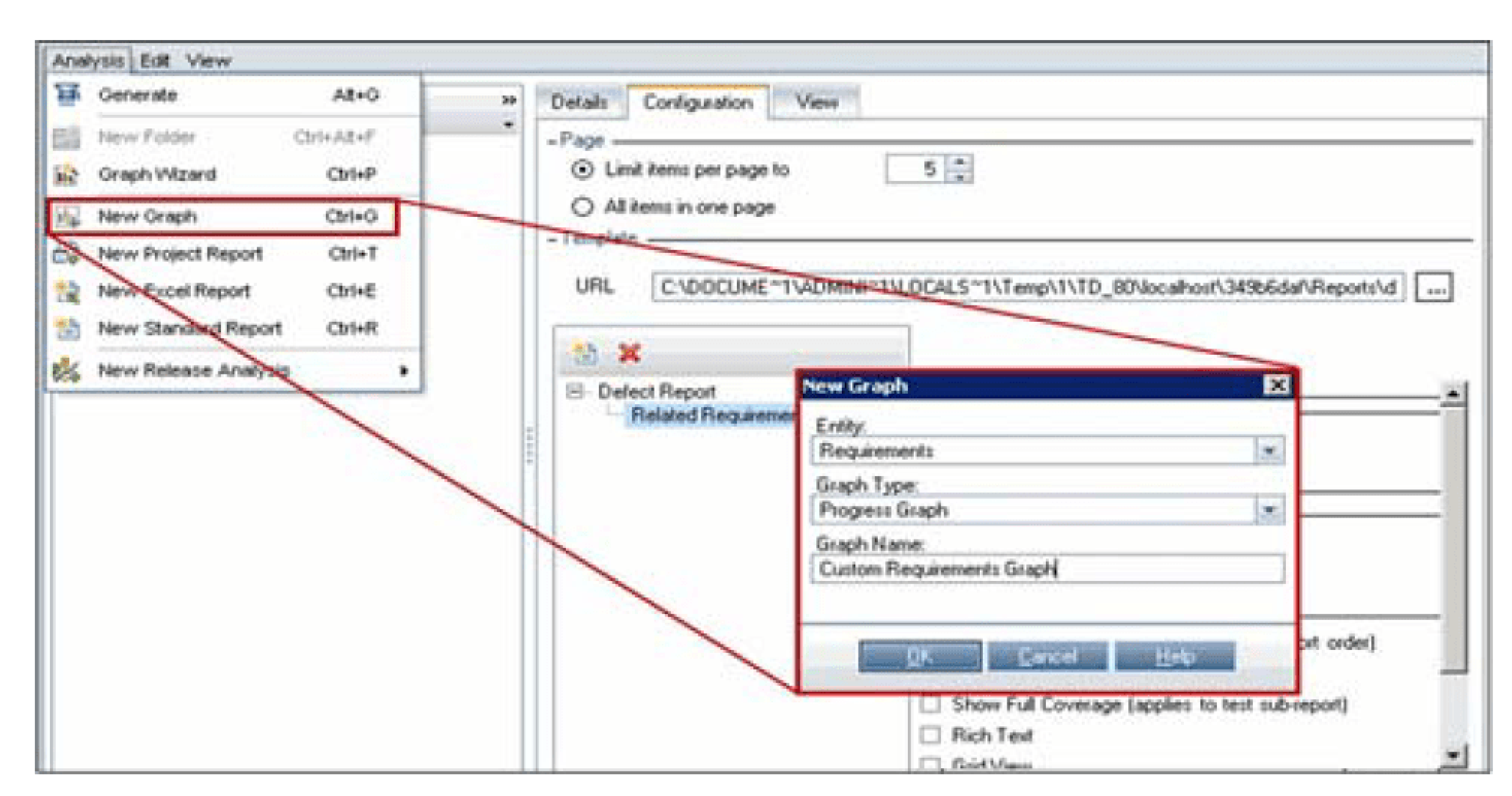


图 16-16　不使用向导创建图表

表 16-5　新图表字段和描述列表

字　段	描　述
Entity	表示图表实例的数据集
Type	表示分析条目类型
Sub Type	表示图表类型
Name	图表名称
Last Modified	表示上次修改图表的日期和时间
Modified By	表示上次修改图表的用户
Owner	代表创建图表的用户。仅仅限于有权限修正公共图表的所有者
描述	图表描述

2. 配置图表

可以定义图表中显示的数据内容，以及数据是如何组织的。配置图表的步骤如下：

1) 在分析树，选择要配置的图表。

2) 单击 Configuration 标签。

3) 配置图表设置。

图 16-17 说明创建图表配置，表 16-6 是配置字段和描述列表。

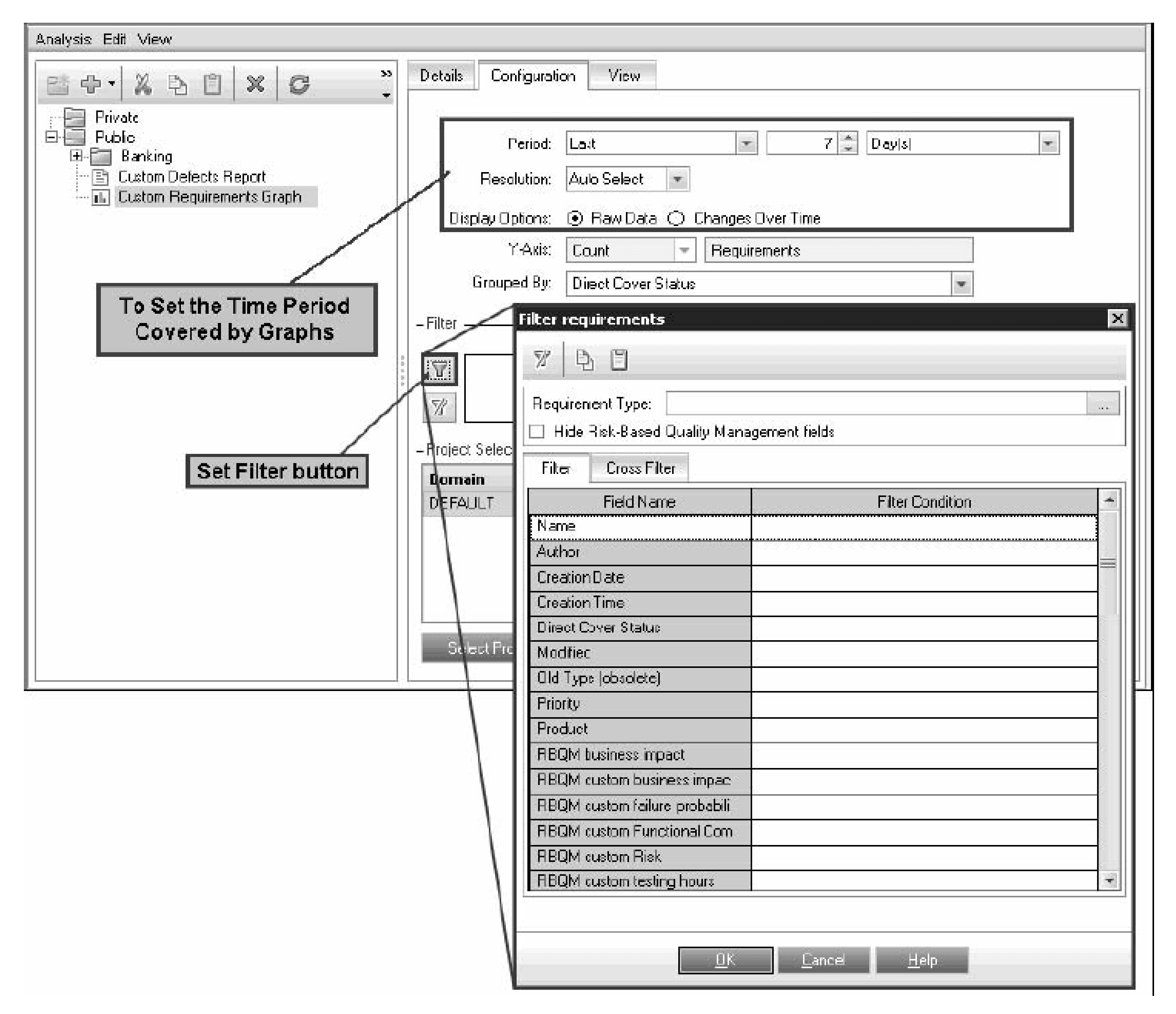


图 16-17　说明创建图表配置

表 16-6　配置字段和描述列表

设　置	描　述
Resolution	可用选项为：Day/Week/Month/Year/Auto Select
Display Options	选择 Regular 查看选择的整个时间段的需求、测试和缺陷数据。选择基于 Time 的 Changes 查看选择的整个时间段的需求、测试和缺陷变化数量。记录以 0 开始
Grouped By	选择字段确定 ALM 在图表中分组数据的信息。可以仅仅使用字符串或列举字段分组数据。提示：对于交叉工程图表，选择 ALM 工程按工程分组数据
Period	选择要图表显示的时间时段
Set Filter/Sort	设置数据过滤/排序
Project Selection	允许单个或多个工程选择
X-Axis	选择字段沿着图表的 X 轴定义信息显示。提示：对于交叉项目图，选择 ALM Projects 显示数据
Y-Axis	在 Defects 图中，可以选择在 Y 轴显示的数据。选择 Count 显示条目总数(例如，启动的缺陷量)。选择 Sum 并选择一个数字字段。例如，选择 Estimates Fix Time 显示修改缺陷需要的时间评估。选择 Actual Fix Time 显示花费在修正缺陷的实际时间。(仅仅在 Defects 图表可用)

16.3　图表窗口元素

通过图表窗口元素能够自定义图表(如图 16-18 所示)，你能够：

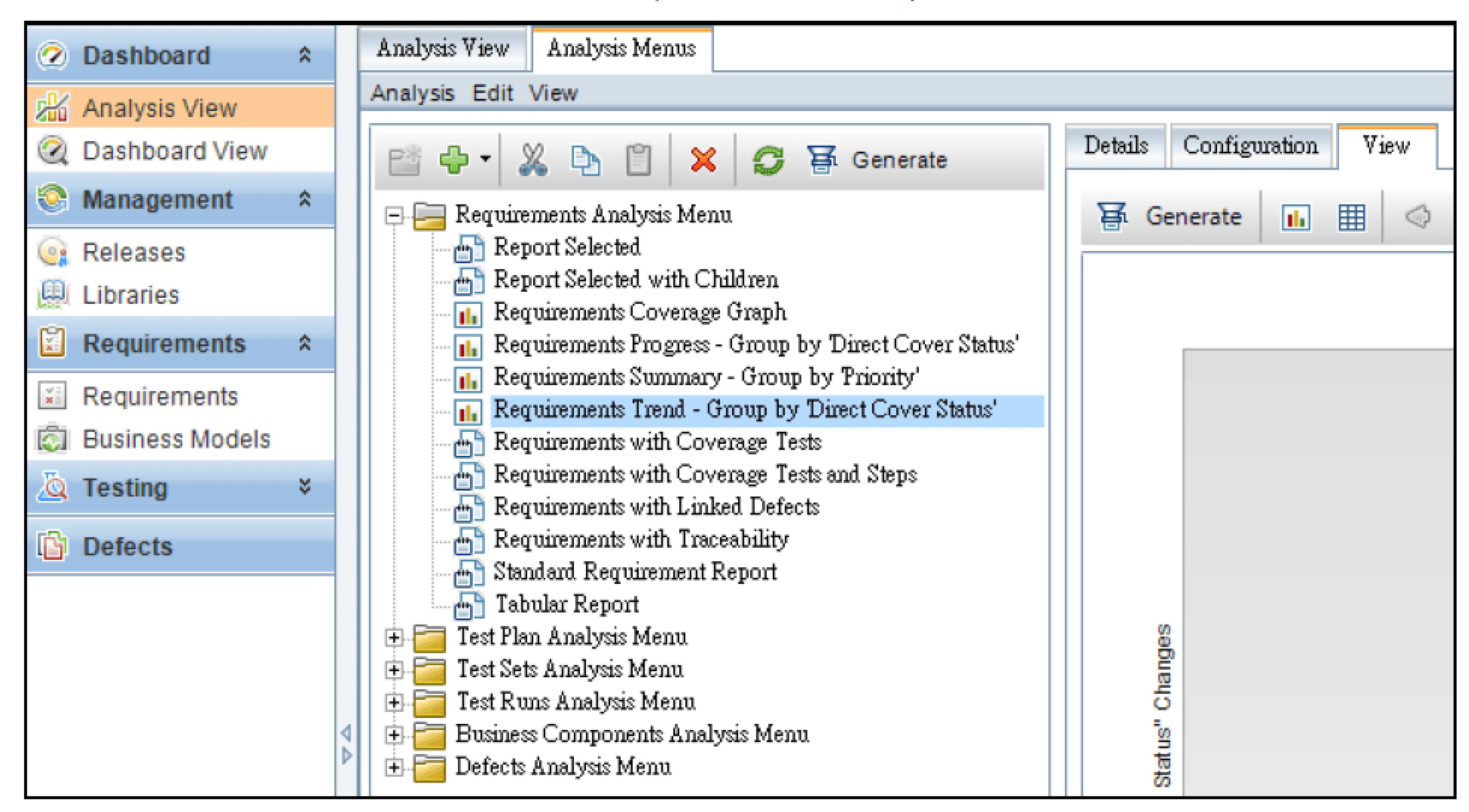


图 16-18　图表窗口元素

(1) 使用 Save 按钮保存图表。

(2) 使用 Set Graph Appearance 按钮修改图表布局。

(3) 使用 Copy Graph To Clipboard 和 Print Graph 按钮重用图表。

(4) 使用 Edit Categories 按钮选择标绘和组织在图表中的数据，或者使用窗口右边的选项改变 X 轴、Y 轴和数据组图标设置。

(5) 使用 Full Screen View 按钮查看图表的放大显示。

单击 Line Chart 按钮查看线形图，单击 Data Grid 在网格格式中查看数据。

创建图表之后，可以基于图表显示数据(如图 16-19 所示)。

(1) 在 Analysis View 中单击 Summary 图。

(2) 单击 View 标签。

(3) 单击 Data Grid 按钮，显示图中汇聚了所有数据的网格。

(4) 要挖掘指定值的详细内容，从网格表中单击值。Drill Down Results 窗口打开，列举代表的具体值条目。

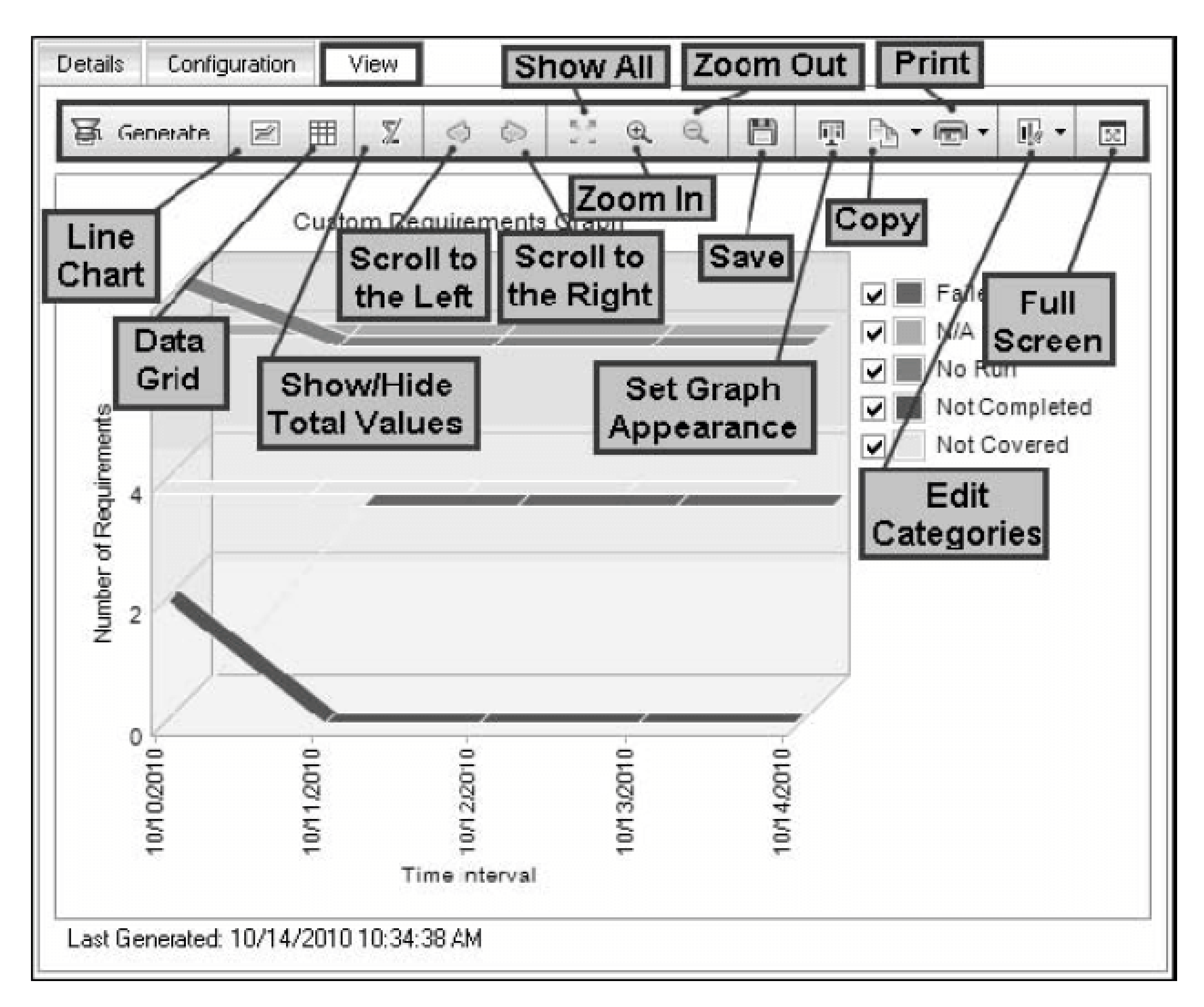


图 16-19　显示图表数据

注意：有个交替方式显示 Drill Down Results 窗口。单击 Bar Chart 或 Pie Chart 标签，单击图表中的分割线。弹出 Drill Down Results 窗口。

16.4　Dashboard 视图

16.4.1　Dashboard 视图

在面板中，可以创建、查看和管理图表、标准报表和 Excel 报表，对于分析 ALM 数据，也可创建面板页，并行显示多个图表。面板视图如图 16-20 所示。

Dashboard 包含分析条目树和面板页树。每个树由 Private 和 Public 根文件夹组成。在每个根文件夹中设置独立的树。公共文件夹中创建的分析条目或面板页所有用户都能访问。私有文件夹中创建的分析条目或面板页只有创建用户可以访问。公共面板页中只能包含公共图表。

根据数据针对用户组的隐藏定义，公共文件夹中的分析条目和面板页会对不同的用户显示不同的结果。

1. 创建 Dashboard 页

创建一个面板页如图 16-20 所示。在面板页，能够在一个页面上排列和查看多个图表。在分析树中选择要放到面板页中的图表。在页面中以喜欢的顺序排列图表，且能够扩大或缩小它们的尺寸。每页最多有 8 个图表，但是能够添加需要的任意数量的图表。实际上每页 4 个图表看上去效果最好，这样也不需要滚动页面了。

面板页只能基于图表创建，即使拥有 Excel 文件夹也不能基于报表或客户自定义查询创建。

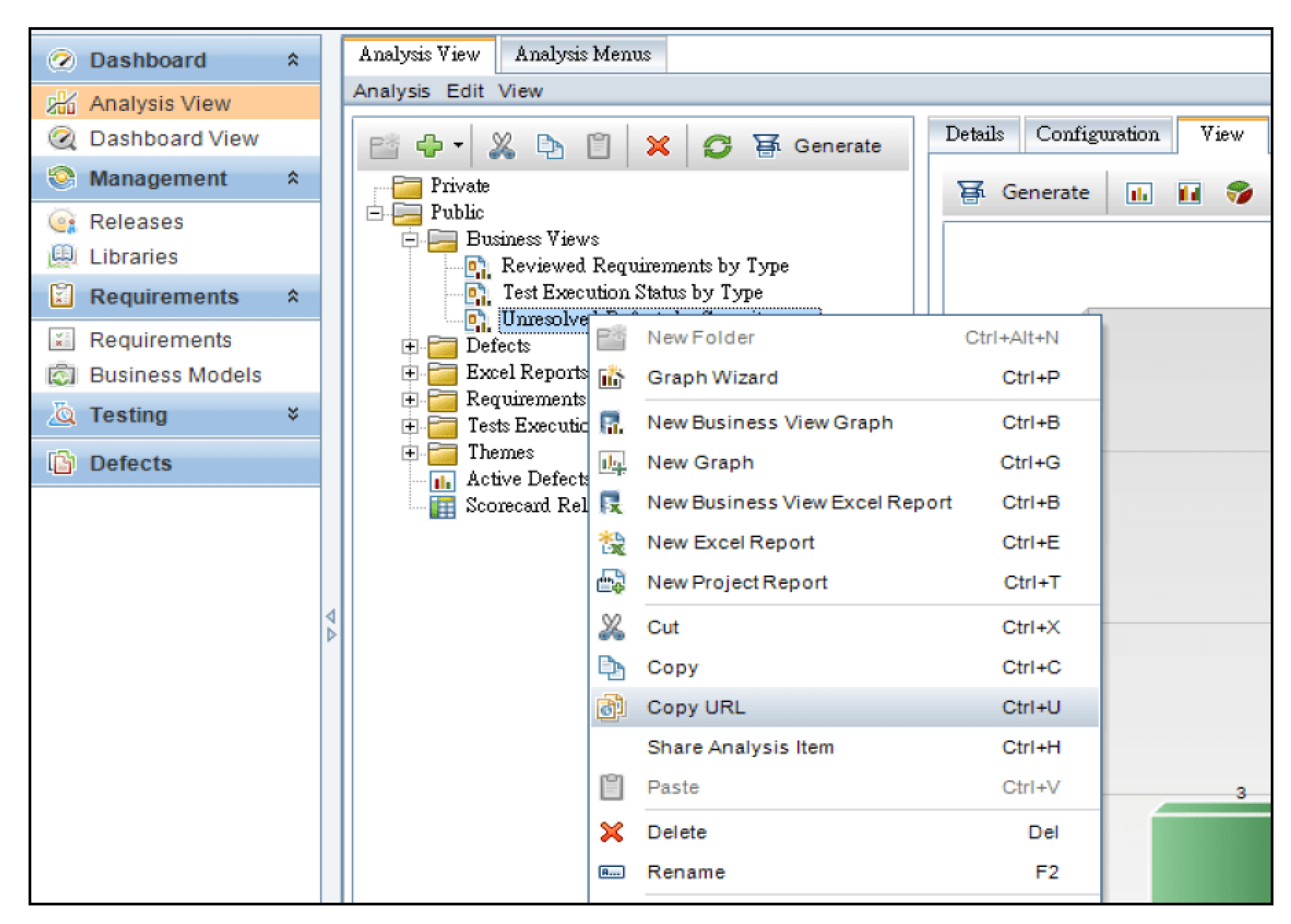


图 16-20　Dashboard 视图

1) 在 Dashboard，单击 Dashboard View 模块。

2) 在面板树，选择一个公共或私有文件夹。

3) 单击 New Page 按钮。或者选择 Dashboard | New Page。

4) 弹出 New Dashboard Page 对话框。输入面板页名字，单击 OK。面板页名称不能包含以下字符：\ ^ *。

2. Dashboard 页添加到选中文件夹下的面板树中

1) 单击 Details 标签，Details 标签显示表 16-7 所列的字段。

2) 选择并排列要加到面板页的图表。

3) 单击 View 标签查看 Dashboard 页。如图 16-21 所示。

表 16-7　Details 字段和描述

字　段	描　述
Name	面板页名字
Last Modified	表示上次修改面板页的日期和时间
Modified By	表示上次修改面板页的用户
Owner	表示创建面板页的用户。仅限于有权限修正公共页面的所有者
Title	表示显示面板页视图的标题
Description	面板页描述

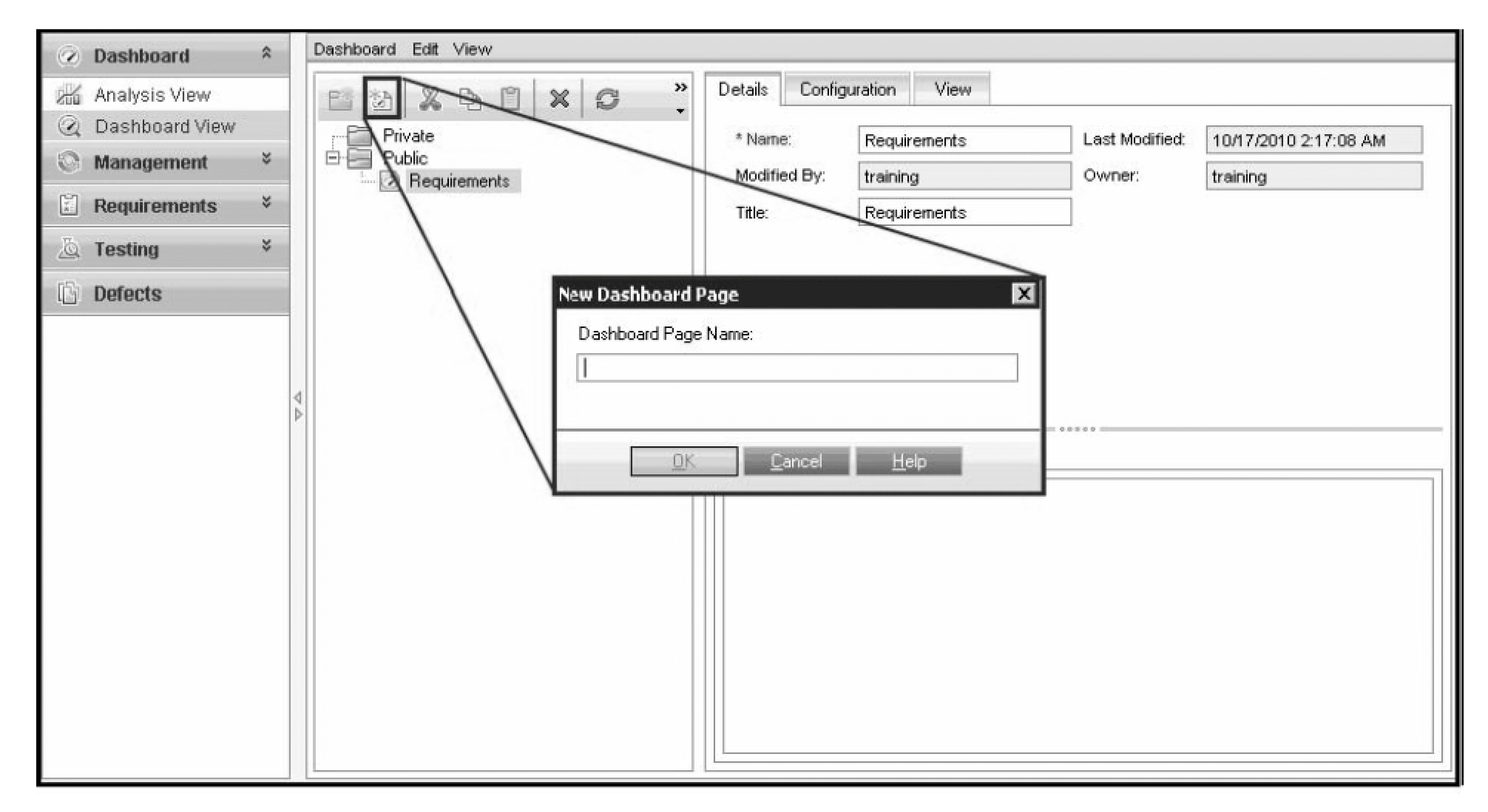


图 16-21　Dashboard 页

Title 表示显示面板页视图的标题，Description 表示面板页描述。

16.4.2　面板配置指导

按照以下给出的指导，创建和配置面板页：

(1) 只有图表能够添加到页面中。

(2) 不能添加报表或自定义查询。

(3) 默认每页添加四个图表。

(4) 公共页面只允许添加公共图表。

(5) Default Graphs Per Page is Four：可以创建至少带有四个图表的面板页。

(6) Only Graphs Can Be Added To a Page：面板页只能用来添加图表。

(7) No Reports or Custom Queries：不能往面板页添加报表。不允许在面板页写自定义 SQL 查询。

(8) Public Pages Allow Only Public Graphs：使用面板可以创建 Public 和 Private 页面。面板公共页面只能添加公共图表，反之亦然。

16.4.3　配置 Dashboard 页

通过在页面上选择和排列图表配置面板页，如图 16-22 所示。面板页的每一行包含一到两个图表。在公共面板页，只能包含公共图表。

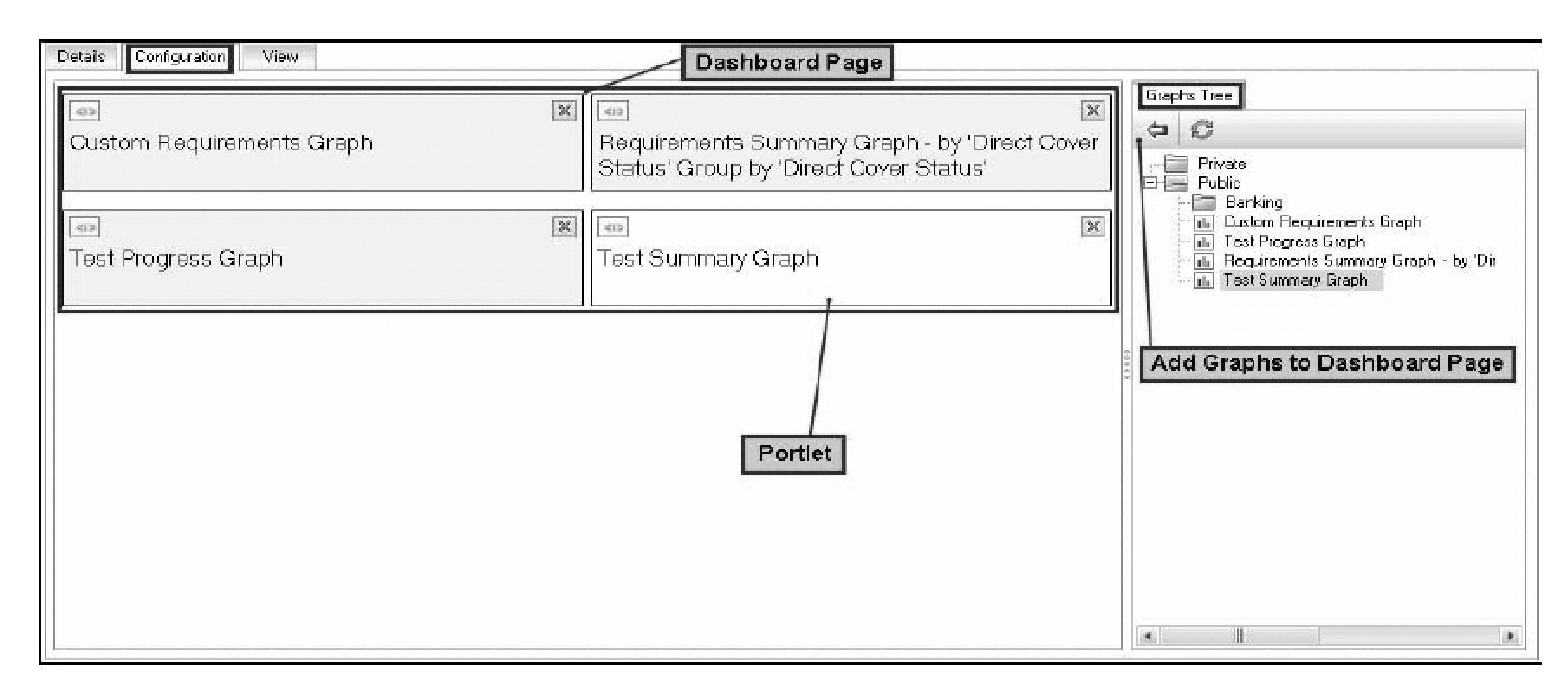


图 16-22　配置面板页

注意：通过设置 Dashboard_Page_Item_Limit 的参数可以改变 ALM 中允许包含在一个面板页中的图表的最大数量。

(1) 在面板中，单击 Dashboard View 模块。

(2) 在 Dashboard 树，选择要配置的 Dashboard 页面。

(3) 单击 Configuration 标签，打开 Select Graphs 面板。

(4) 要刷新图表树，单击 Refresh 按钮。

(5) 选择一个图表，单击 Add Graph to Dashboard Page 按钮。也可以将一个图拖拽到已存在的图的上边或下边的新行，或图旁边的空框中。显示图表标题的占位符标签在 Configuration 生成。

注意：公共面板页中不能包含来自私有分析文件夹的图表。

(6) 将图表向侧边、上边或下边移动，选择并拖拽占位符到新位置。图表间不能遗留空行。

16.4.4　查看面板页

查看面板页的操作参见图 16-23。

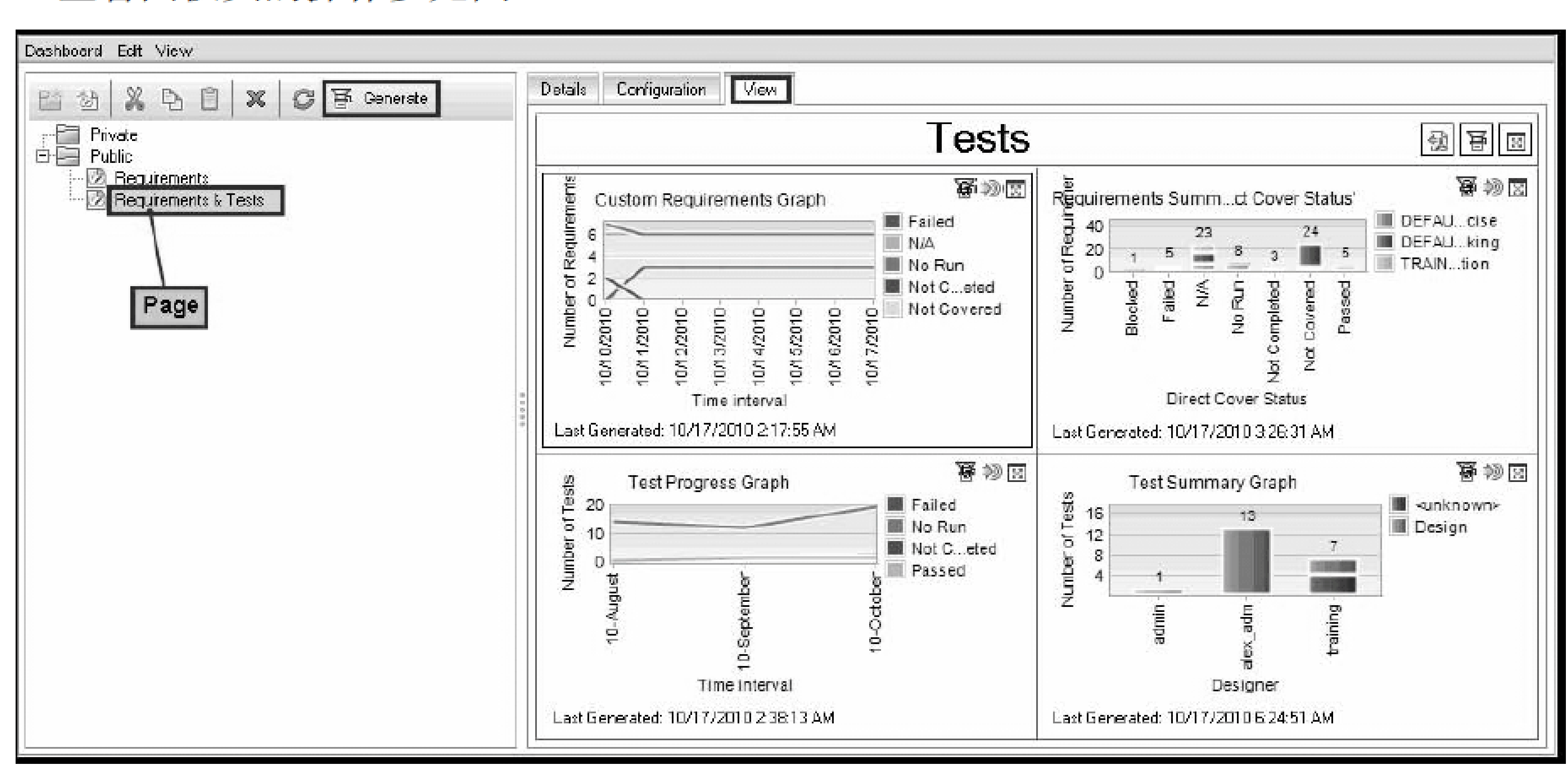


图 16-23　查看面板页(Dashboard View Page)

16.4.5　查看面板选项

查看面板选项，如图 16-24 所示：

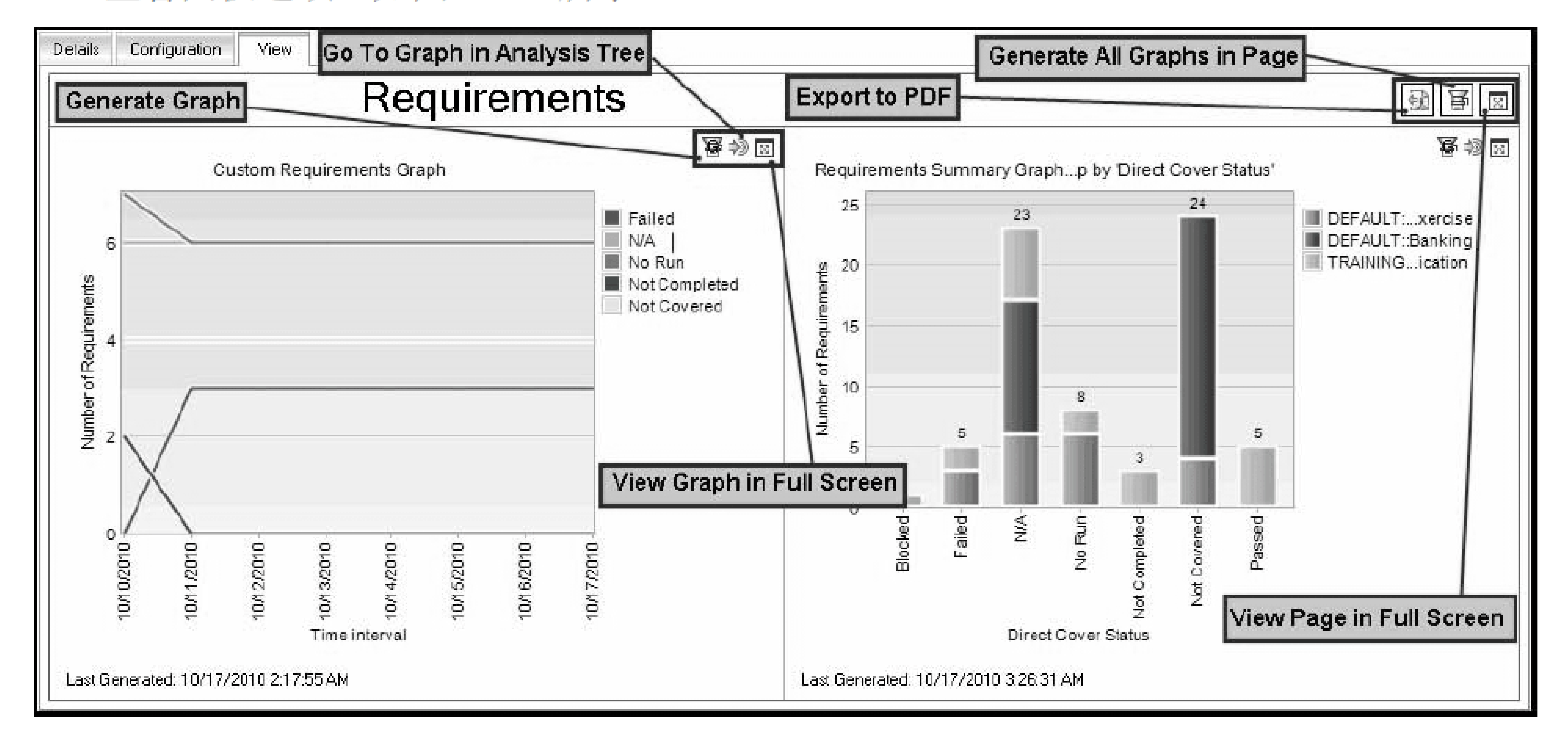


图 16-24　面板选项(Dashboard View Option)

(1) 在柱形图或饼形图中，可以根据图表的局部内容所代表的记录进行深入探讨。

(2) 要在全屏模式下查看面板页，单击位于图表右下角的 View Page in Full Screen 按钮。要返回标准视图，单击页面右上角的 close 按钮。

(3) 要以全屏模式查看一个页面，单击图表右下角的 View Graph in Full Screen 按钮。使用工具条按钮能够调整图表显示，保存或打印图表。下次生成面板页时所作的图表显示调整会被重设。要返回到面板页视图，单击位于图表右上角的 Close 按钮。

(4) 要刷新所有图表的数据，单击位于页面右上角的 Generate All Graph in Page 按钮。要刷新一个图表的数据，单击位于图表左上角的 Generate Graph 按钮。ALM 刷新图表并更新最新的 Generated 时间和日期。

(5) 要导航到分析树中的一个图表，单击位于图表左上角的 Go To Graph in Analysis Tree 按钮。或者，双击图表。Analysis View 打开，分析树中的图表被选中。

16.4.6　Dashboard 细节

查看面板详细内容，如图 16-25 所示：

(1) 以饼形图或柱形图格式显示图表。

(2) 单击一个分割或一个柱条，如果这个分割或柱条代表目前正工作的工程记录，弹出 Drill Down Results 窗口。

(3) 在 Drill Down Results 窗口，单击右上角的 Select Columns 按钮，定义网格表中显示哪些列，以及这些列的显示顺序。

(4) 双击记录按钮打开记录的详细内容。<Module>Details 对话框打开。在对话框中编辑记录详细内容。

(5) 单击 OK 或 Cancel 关闭对话框，返回 Drill Down Results 对话框。

(6) 关闭 Drill Down Results 对话框返回 View 标签。

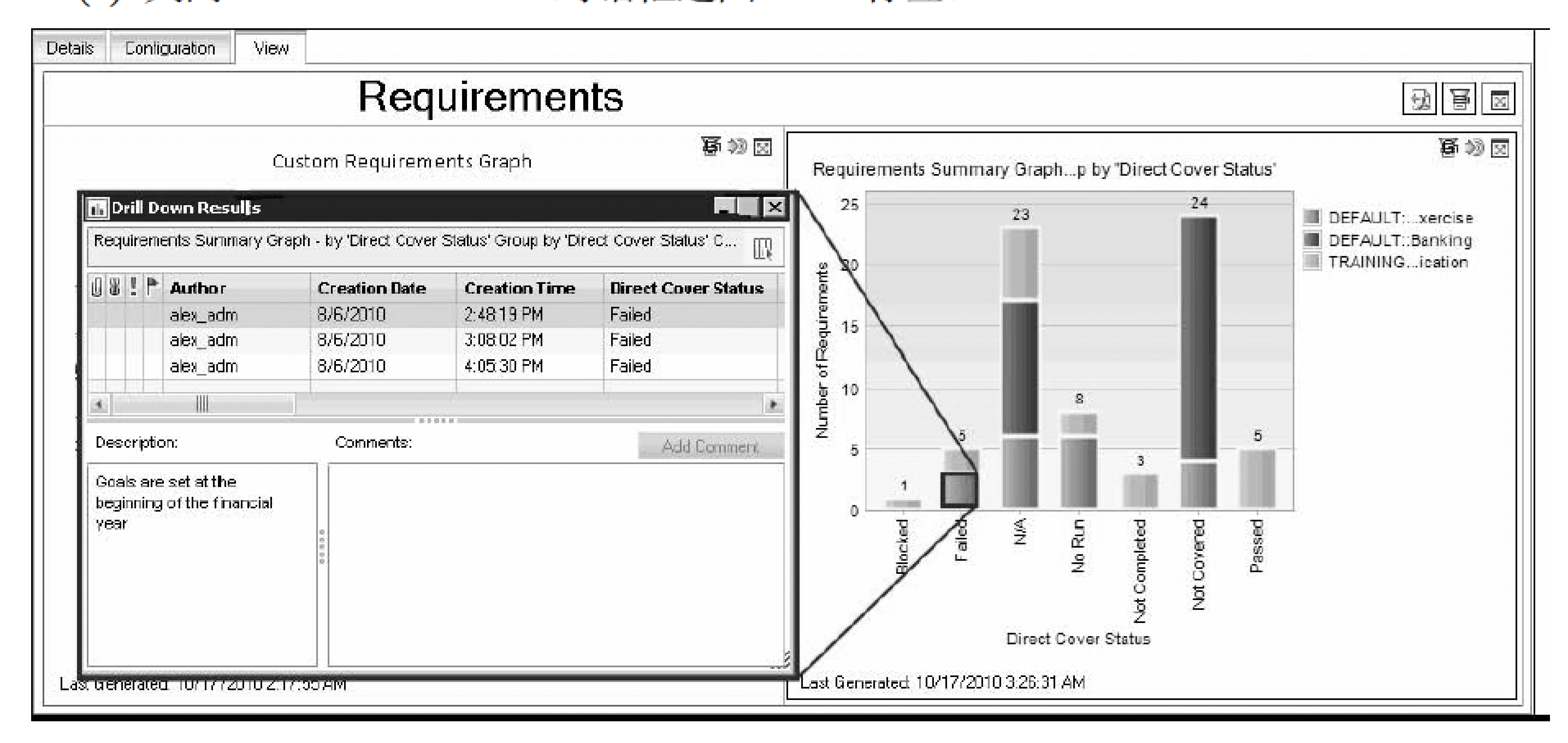


图 16-25　面板详细页

习题与思考题

1. 在哪种 ALM 模式下能够生成图表？
2. ALM 的 5 种不同图表类型有哪些？
3. 列举 Document Generator 工具的一种用途。

练习：报表和分析

完成 Flight Reservation 程序的测试之后，需要生成多种报表和图表进行以下分析：

1. 工程中创建的需求总量
2. 检查过的需求数量
3. 创建的测试的状态
4. 工程的当前状态
5. 需求覆盖数据的范围

在练习中，要完成以下任务：

第 1 部分：生成分析报表和图表

第 2 部分：生成 Project 报表

第 3 部分：使用 Dashboard

第 4 部分：使用 Dashboard Configuration View

(该练习的指导步骤请参见本章正文)

第Ⅲ部分　HP ALM 工具提高篇

本部分共有 4 章内容，主要是 HP ALM 工具的使用提高。

第17章　需求风险分析

17.1　基于风险的质量管理概述

一般情况下，当计划进行需求测试时，可使用资源都是有限而且不可能完全地测试每一项需求。所以测试需求时必须选择折中方案，对那些低风险区域只进行部分测试。基于风险的质量管理特性有助于通过需求的性质和可用的资源来计算测试每项需求的级别。可以根据这些推荐的规范计划测试过程。每一种基于风险的质量管理的需求类型都可以支持被称为分析需求的风险分析或者是被称为评估需求的单个风险评估。

分析需求是一种代表需求树层次中高级别形式的需求，比如文件夹类型。在需求树中以评估需求为基础，在分析需求上执行风险分析。

1) 多种评估需求的风险结果的集成产生整体风险分析，它可以用来确定测试工作和测试策略。

2) 评估需求是分析需求的子集并且处于需求树层次中较低级别形式的需求。在特殊分析需求下的评估需求组成了风险分析的基础部分。对于每一个分析需求下的评估需求都可以计算出风险和功能复杂度。

17.2　如何跟踪需求

跟踪需求描述如何定义需求之间的跟踪链，以及如何查看需求之间的关联和依赖关系。

17.2.1　定义跟踪链

(1) 在 Requirement 模块中，选择 Requirement Details，单击 Requirement Traceability。

(2) 定义一个跟踪链，从需求树中选择一个需求。

(3) 在 Relationships 选项卡中，单击 Add Requirement Traceability 按钮，需求树在右边面板中显示，添加跟踪链。

17.2.2　查看跟踪影响

单击 Impact Analysis 选项卡。查看需求间的关联和依赖关系。

17.2.3　生成跟踪矩阵

生成跟踪矩阵以完善需求间的关系完整性。

在需求模块，选择 View | Traceability Matrix。配置跟踪矩阵。

17.2.4　添加需求追踪

当在需求树中创建一个需求后，可以在各种需求之间建立追踪关系，如图 17-1 所示。需求追踪矩阵是 ALM 的一个特性，它有助于在多种需求之间建立联系。当分析需求变更所带来的影响时，跟踪链可以帮助识别其他需求可能受到的影响。

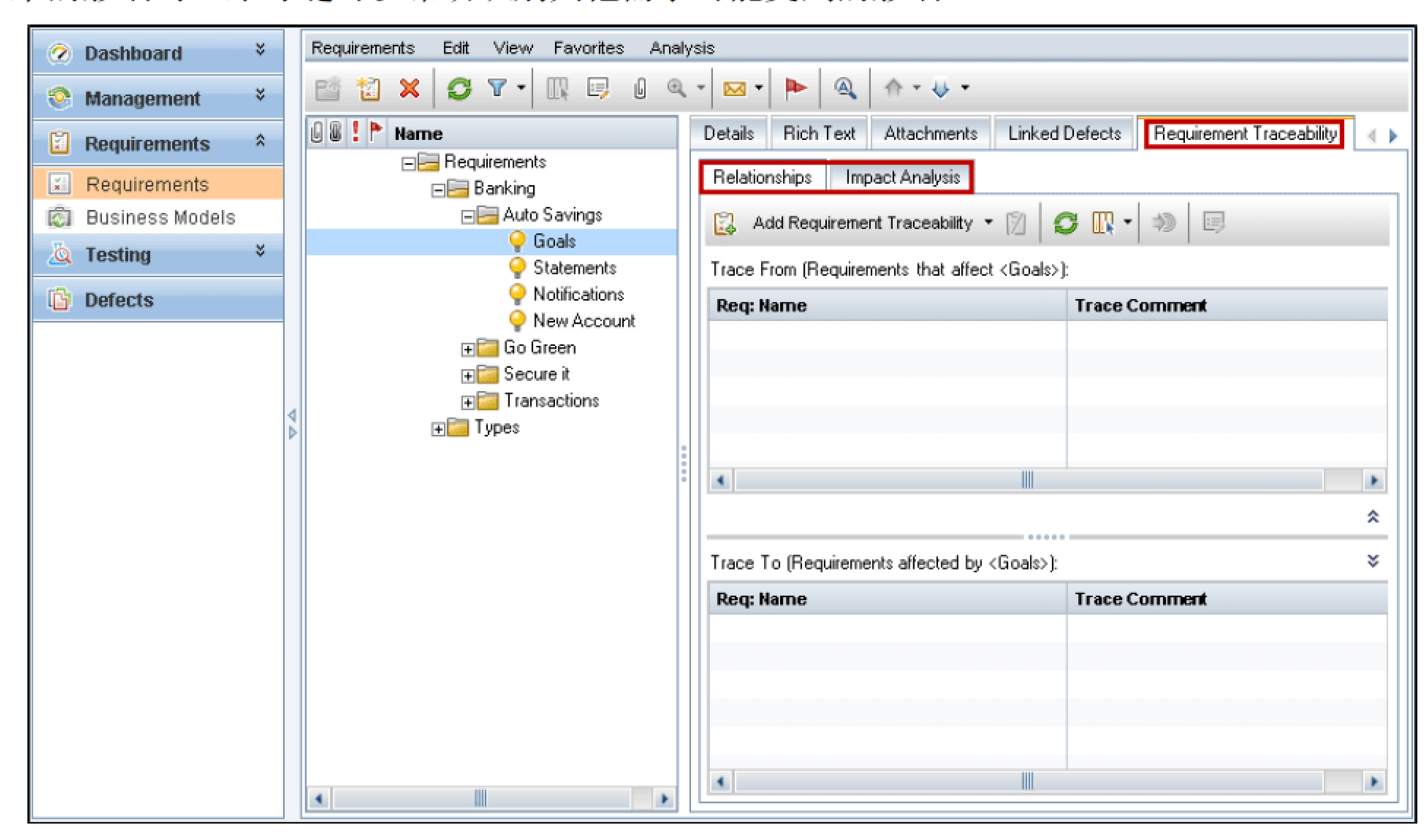


图 17-1　创建需求并建立需求追踪关系

可以运用详细需求视图添加两个需求之间的跟踪链。在详细需求视图右侧，单击 Requirement Traceability 选项卡。

需求追踪标签为跟踪链提供了如下可用选项：

1) Relationships：添加和删除需求之间的跟踪链。

2) Impact Analysis：通过查看需求关系分析需求变化带来的影响。

可以运用关系视图标签查看存在于需求间的跟踪链。另外，关系视图标签实现在需求中添加和删除跟踪链。关系视图标签为跟踪链的有效性提供了跟踪的来源和跟踪的趋势网格。跟踪来源网格显示需求树中影响需求选择的需求。例如：图 17-2 表明了付款方式需求受到取消保留需求变更的影响。

添加需求追踪的步骤如下：

(1) 在 Relationships 工具条上，单击 Add Requirement Traceability 按钮，需求树出现在页面右侧，如图 17-3 所示。

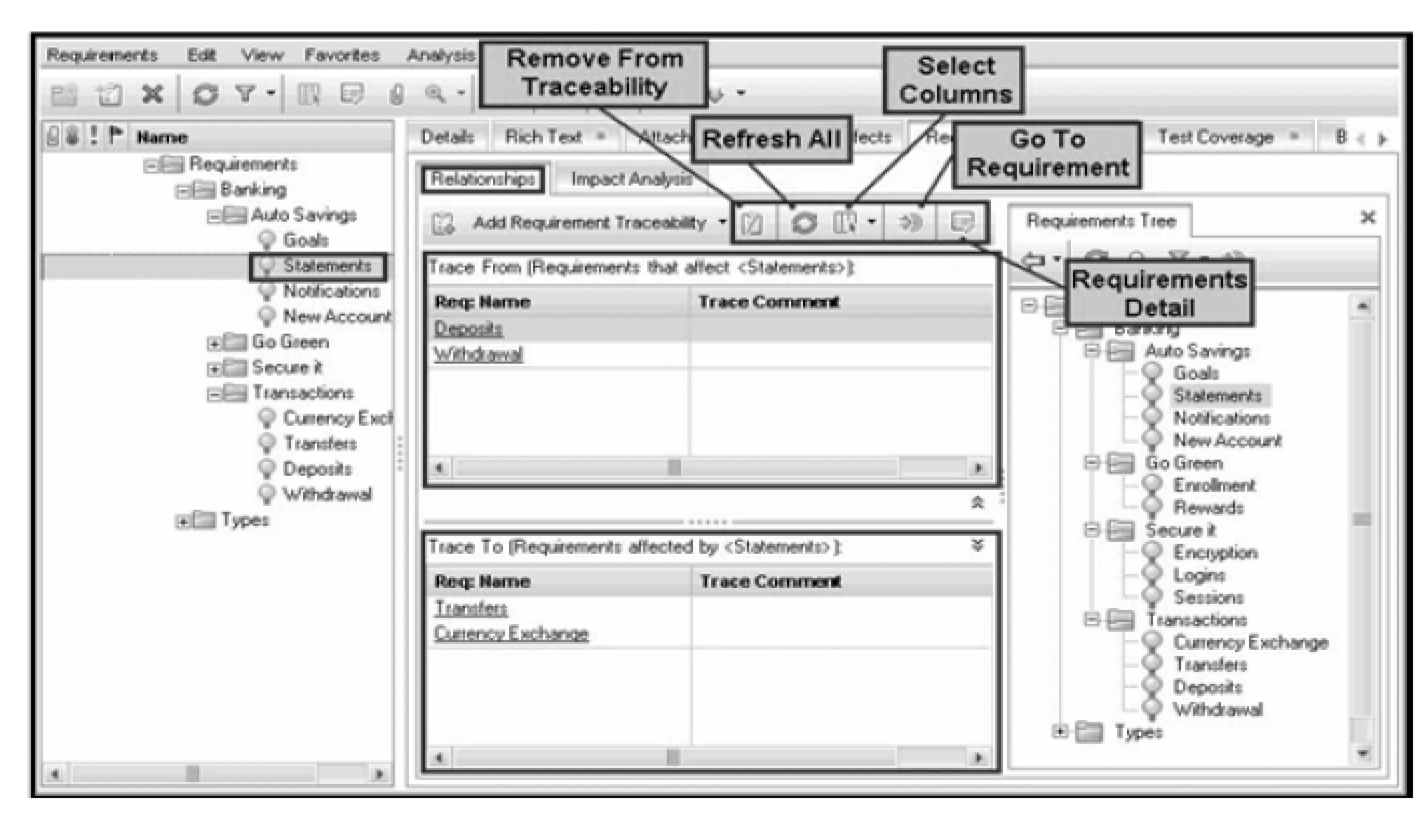


图 17-2　需求关系视图

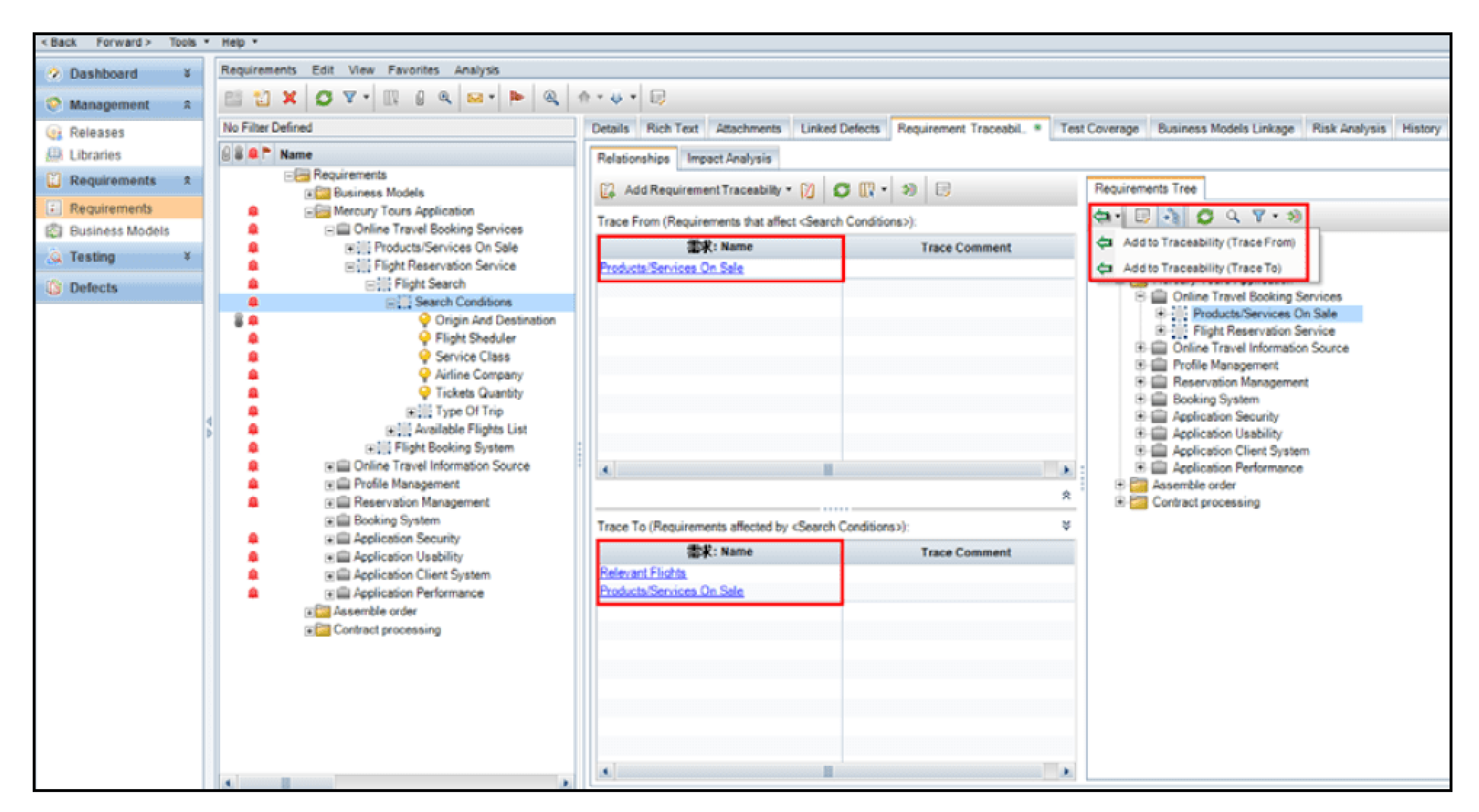


图 17-3　用关系选项卡添加需求跟踪链

(2) 在需求树中，展开 Requirements 并选择一项需求。

(3) 将选择的需求添加到 Trace From 列表中，单击 Add to Traceability 箭头并选择 Add to Traceability(Trace From)，被选择的需求出现在 Trace From 网格中。

(4) 将选择的需求添加到 Trace To 列表中，单击 Add to Traceability 箭头并选择 Add to Traceability(Trace To)。被选择的需求出现在 Trace To 网格中。

关系选项卡使你能在需求之间添加跟踪链。

17.3　影响分析

在 Relationships 视图中建立跟踪关系后，在 Impact Analysis 选项卡中，通过回顾跟踪关

系来分析需求变化的影响。

Impact Analysis 视图以层次结构显示需求。而需求树中的每项需求都以图标形式表示，这些图标有利于理解各项需求间的关联和依赖关系。如图 17-4 所示，在 Trace From 树中，Booking System 需求图标表示 Payment Methods 来源于 Booking System 这个父需求系统。这意味着任何对于 Booking System 或其子需求的变更都会影响 Payment Methods 。

同样，如图 17-5 所示，在 Trace To 树中，Airline Companies 需求标志着 Payment Methods 指向 Airline Companies 这个子需求。这意味着任何对于 Payment Methods 需求的变更会影响 Airline Companies 及其子需求。

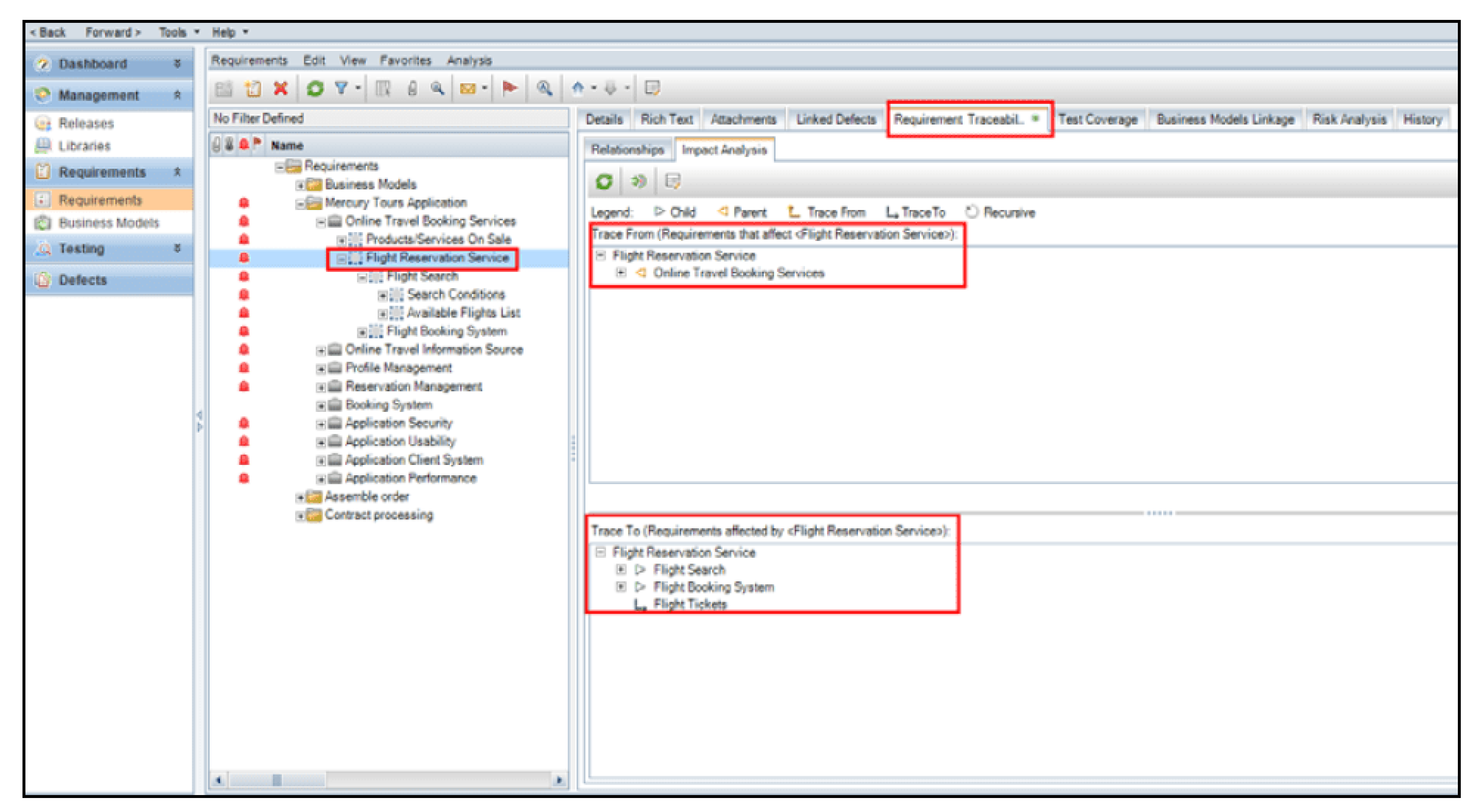


图 17-4　Trace To 树中的需求变化所带来的影响(一)

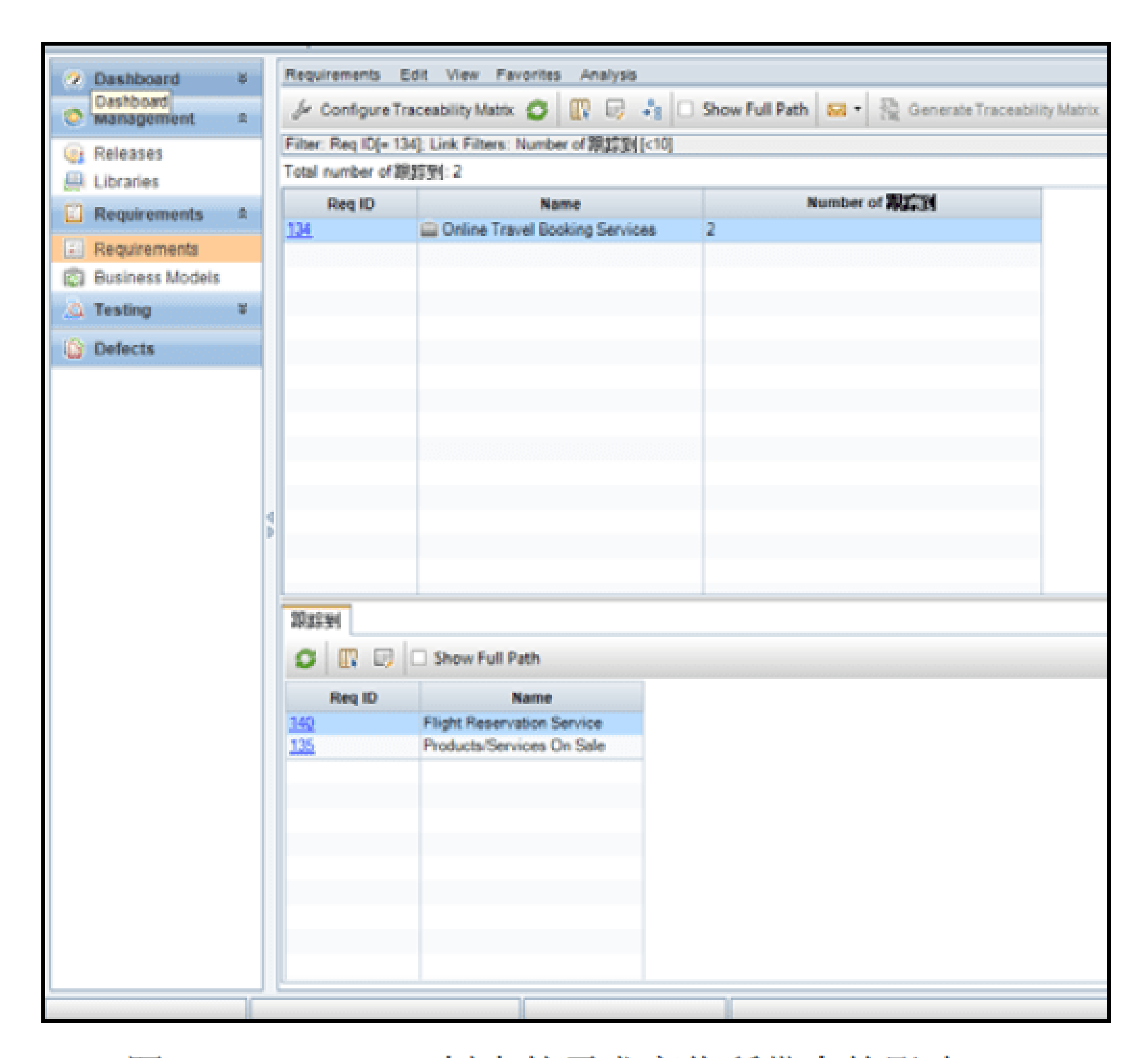


图 17-5　Trace To 树中的需求变化所带来的影响(二)

17.4　跟踪矩阵

跟踪矩阵使你能够明晰需求与需求间及需求与测试间的关系。它能帮助验证是否已满足所有需求，并且可以识别出需求范围的变化情况。

17.4.1　跟踪矩阵列表

1. 原始需求、关联需求和测试。
2. 各原始需求的关联总数。
3. 低值可能意味着原始需求没有与足够的需求和测试关联。
4. 高值可能意味着原始需求过于复杂，可能需要被简化。
5. 零价值意味着没有任何关联。

17.4.2　创建跟踪矩阵

(1) 在 Requirements 模块菜单中，单击 View | Traceability Matrix。

(2) 单击 Configure Traceability Matrix 链接，弹出 Configure Traceability Matrix 对话框。

(3) 单击 Define source requirements 链接，如图 17-6 所示。

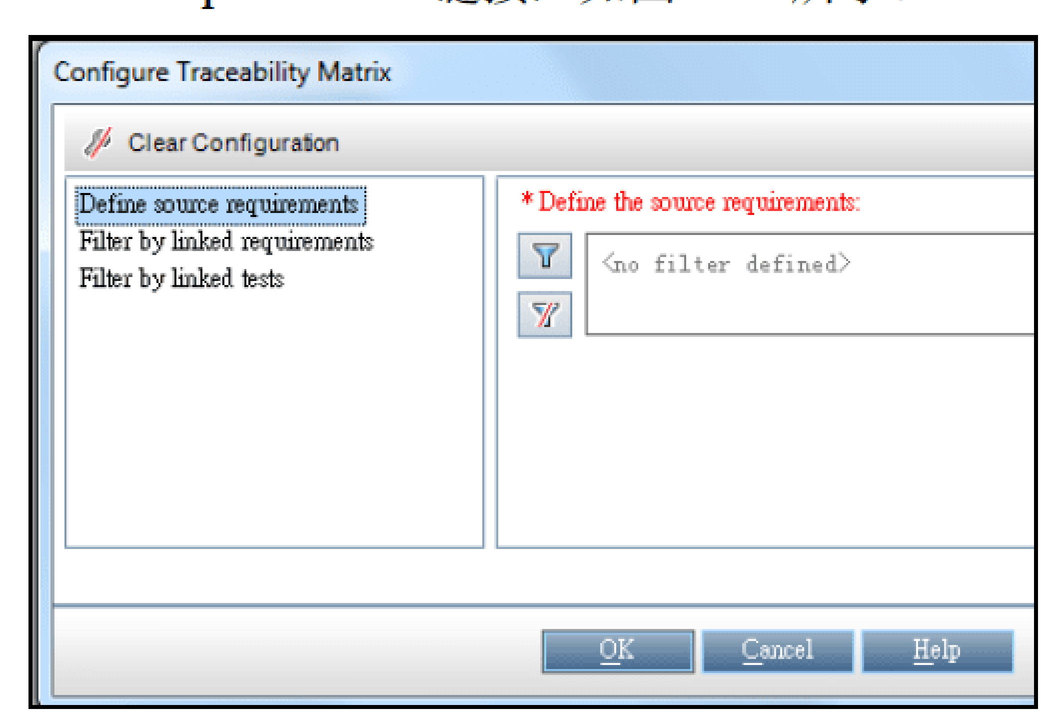


图 17-6　跟踪矩阵的配置——定义初始需求

(4) 单击 Set Filter/Sort 按钮来过滤/分类需求。Filter Requirements 对话框弹出。

(5) 单击 Requirement Type 的下拉菜单或单击过滤条件区域，弹出 Select Filter Condition 对话框。

(6) 在 Select Filter Condition 对话框中输入 Filter Condition， 并单击 OK 按钮。

(7) 在 Filter Requirements 对话框中单击 OK 按钮。

(8) 单击 Filter by linked requirements 链接，如图 17-7 所示。

(9) 选中 Filter by linked requirements 复选框。

(10) 在 Include source requirements 下拉选择框中选中 4 个可选项中的一项：

1) affected by

2) not affected by

3) affecting

4) not affecting

(11) 如果在 Include source requirements 下拉选择框中选择 affecting 或者 not affecting，再从以下三个选项中选择一项：

1) direct children and traced to requirements

2) direct children

3) traced to requirements

(12) 单击 Set Filter/Sort 按钮来为需求设置过滤/排序条件。

(13) 单击 Filter by linked tests 链接。

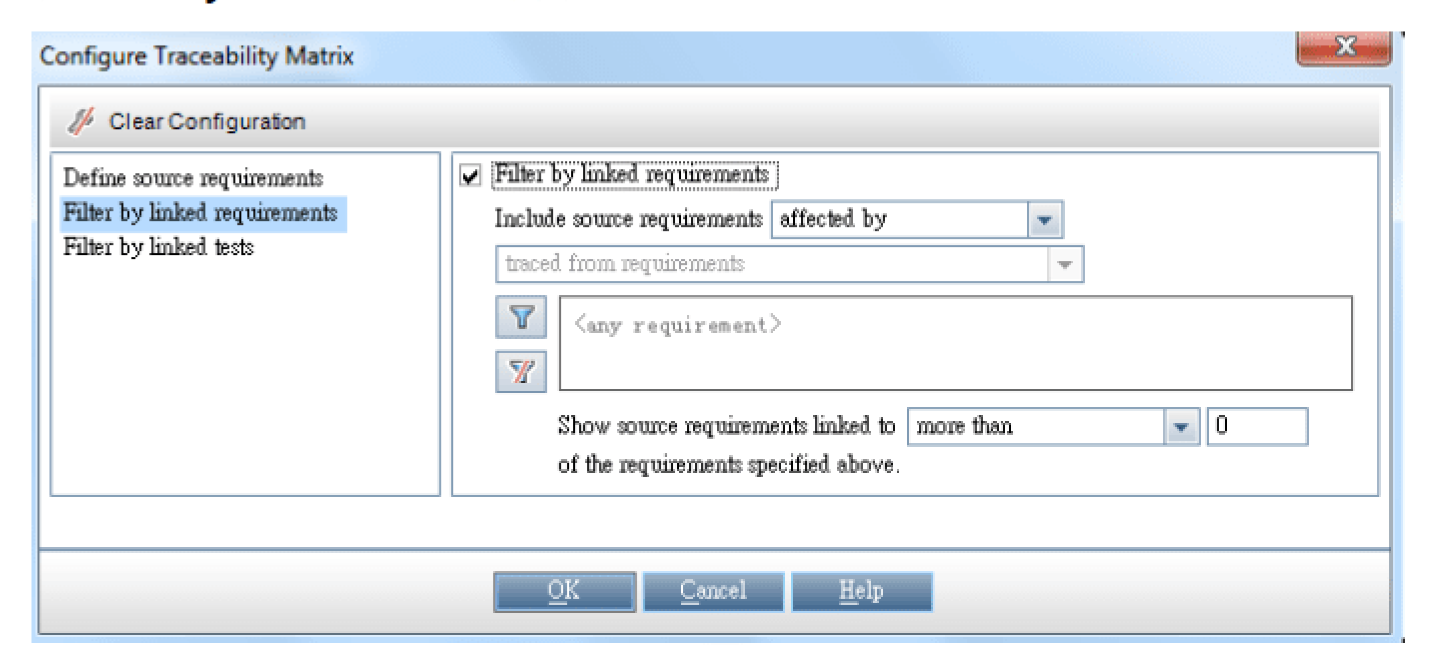


图 17-7　跟踪矩阵的配置——用所链接的需求进行过滤

(14) 选择 Filter by linked tests 复选框。

(15) 选择 Include source requirements Linked To 或者 Not Linked To The Following Tests。

(16) 单击 Set Filter/Sort 按钮来为测试设置过滤/排序条件。

(17) 在 Configure Traceability Matrix 对话框单击 OK 按钮。

(18) 最后查看需求矩阵，如图 17-8 所示。

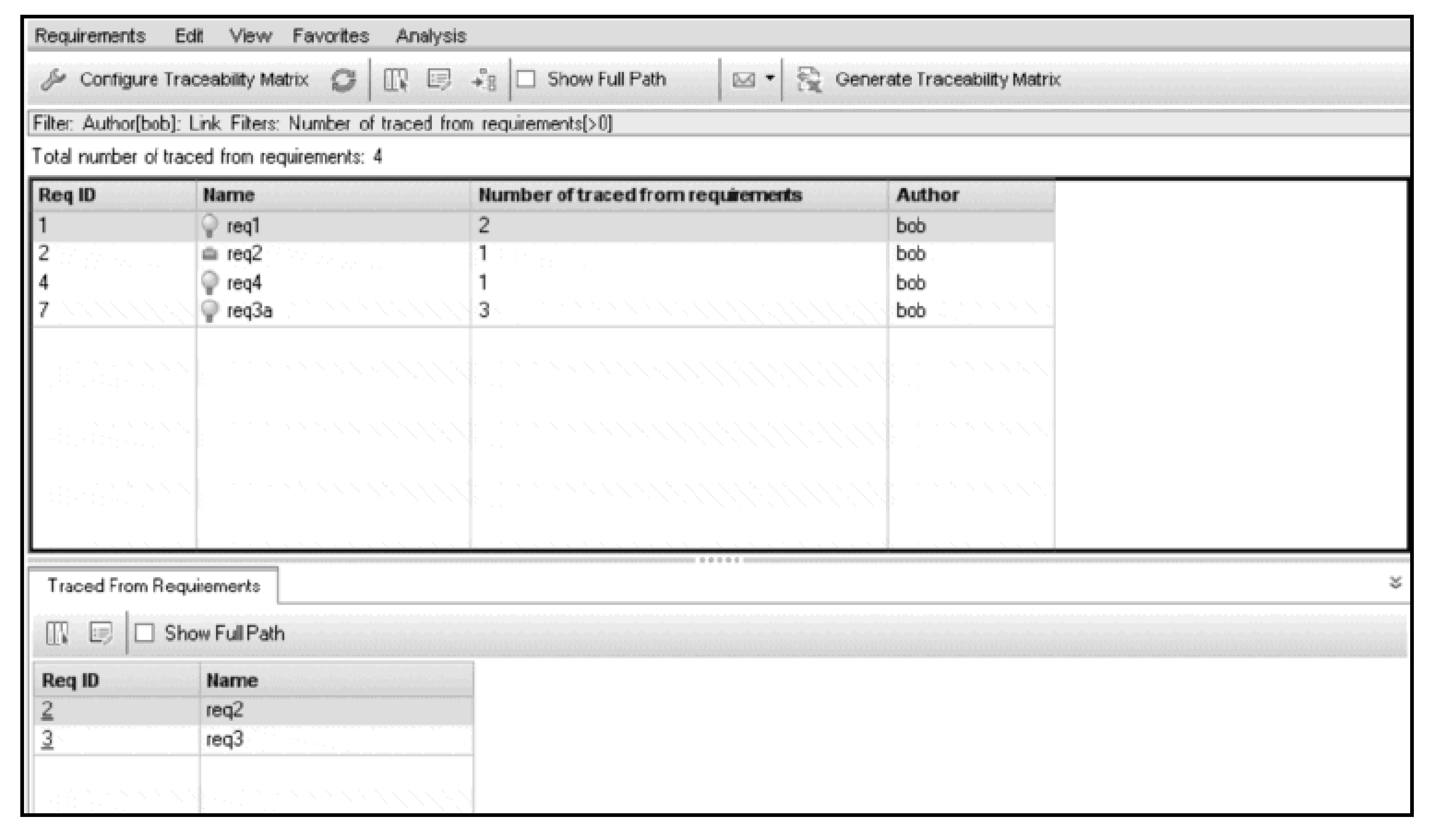


图 17-8　查看需求矩阵

17.5　使用基于风险的测试

在准备对需求进行测试计划时经常遇到挑战，比如资源短缺迫使你妥协只对一些需求进行部分测试。所以必须识别低紧急程度的需求或与其相关的低风险需求。

风险由业务紧急程度和失败几率组成。业务紧急程度标识需求对业务的重要性。失败几率表示基于需求测试的失败的可能性大小。功能复杂度指的是需求实现的复杂性程度。每一个需求类型都可以进行基于风险质量管理。

识别需求的紧急程度和关联到需求的风险后，使用 ALM 基于风险测试的功能计算被测试需求所处的级别，如图 17-9 所示。ALM 定义的四个测试标准为：FULL、PARTIAL、BASIC 和 NONE。基于风险的测试运用需求类型和资源可用性计算测试标准。

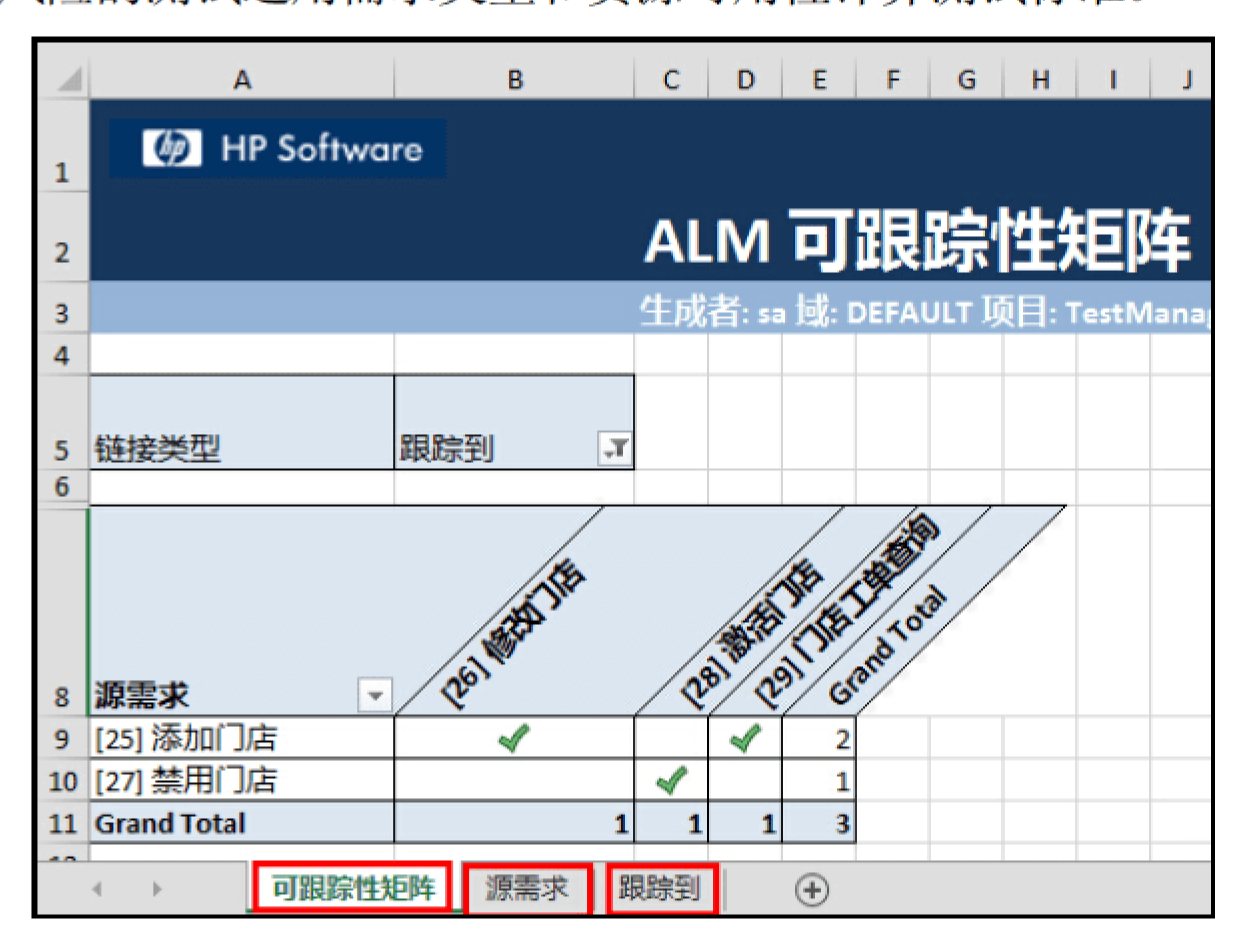


图 17-9　基于需求矩阵的需求测试要求分析

ALM 初始版本：没有基于风险的质量管理功能模块。

风险评估(如图 17-10 所示)包括以下几个步骤：

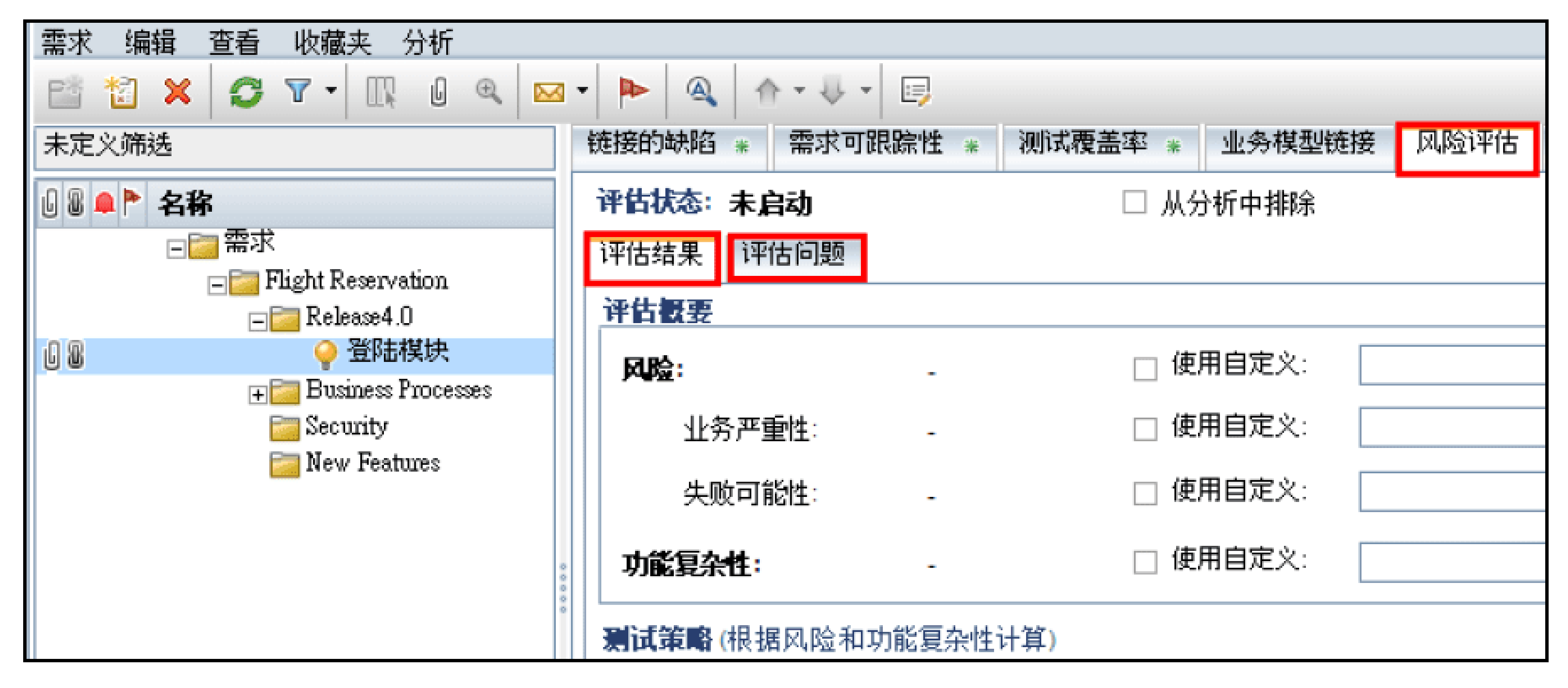


图 17-10　风险评估界面

(1) 评估需求
(2) 定义测试策略设置
(3) 定义测试策略
(4) 分析测试策略

1. 评估需求

1) 展开 Risk Assessment 标签。在 Requirements 模块中，选择 View | Requirement Details。在需求树中的分析需求下面，选择一个评估需求。单击 Risk Assessment 选项卡。

2) 确定风险和功能复杂度。单击 Assessment Questions 标签，会出现以下子标签：Business Criticality、Failure Probability 和 Functional Complexity。在每个子标签，为一系列标准赋值。可以忽略以基础计算得来的值，并用指定的值代替。为每个分类直接赋值，单击 Assessment Results 标签。

3) 对每项在需求分析下的评估需求，指定或者计算风险和功能复杂度。

2. 定义测试策略设置

1) 在需求树中，选择分析需求，单击 Risk Analysis 标签。

2) 在分析常量下，为测试的分析需求和评估需求定义初始的设置。这些设置包括完全测试一个指定功能复杂性的需求的时间和对一个需求执行部分测试或者是基础测试的时间。还要决定每项风险和功能复杂性所需的测试级别。

3. 确定测试策略

1) 计算测试策略。在 Risk Analysis 标签下，单击 Analyze 按钮，计算需求分析下的每个评估需求的测试级别和测试时间。需要的总测试时间、分配的测试总时间和需要的总开发时间将被更新。

2) 调整测试策略。在分析常量下，可以对测试策略进行修改以使有足够的时间执行测试过程，而且没有资源浪费。

3) 应用每个评估需求的结果。单击 Analysis and Apply to Children 按钮，分析结果在分析需求中所有匹配当前过滤条件的评估需求中得到应用。

4. 分析测试策略

1) 生成分析需求测试策略的详细报告，在风险分析视图中，单击 Report 按钮，弹出 Generate Report 对话框。

2) 分析评估需求的测试策略，从需求树中选择一个评估需求，单击 Risk Assessment 标签。结果在 Assessment Results 的子标签中显示出来。

17.6　基于风险的质量管理

如图 17-11 所示，基于风险的质量管理流程如下：

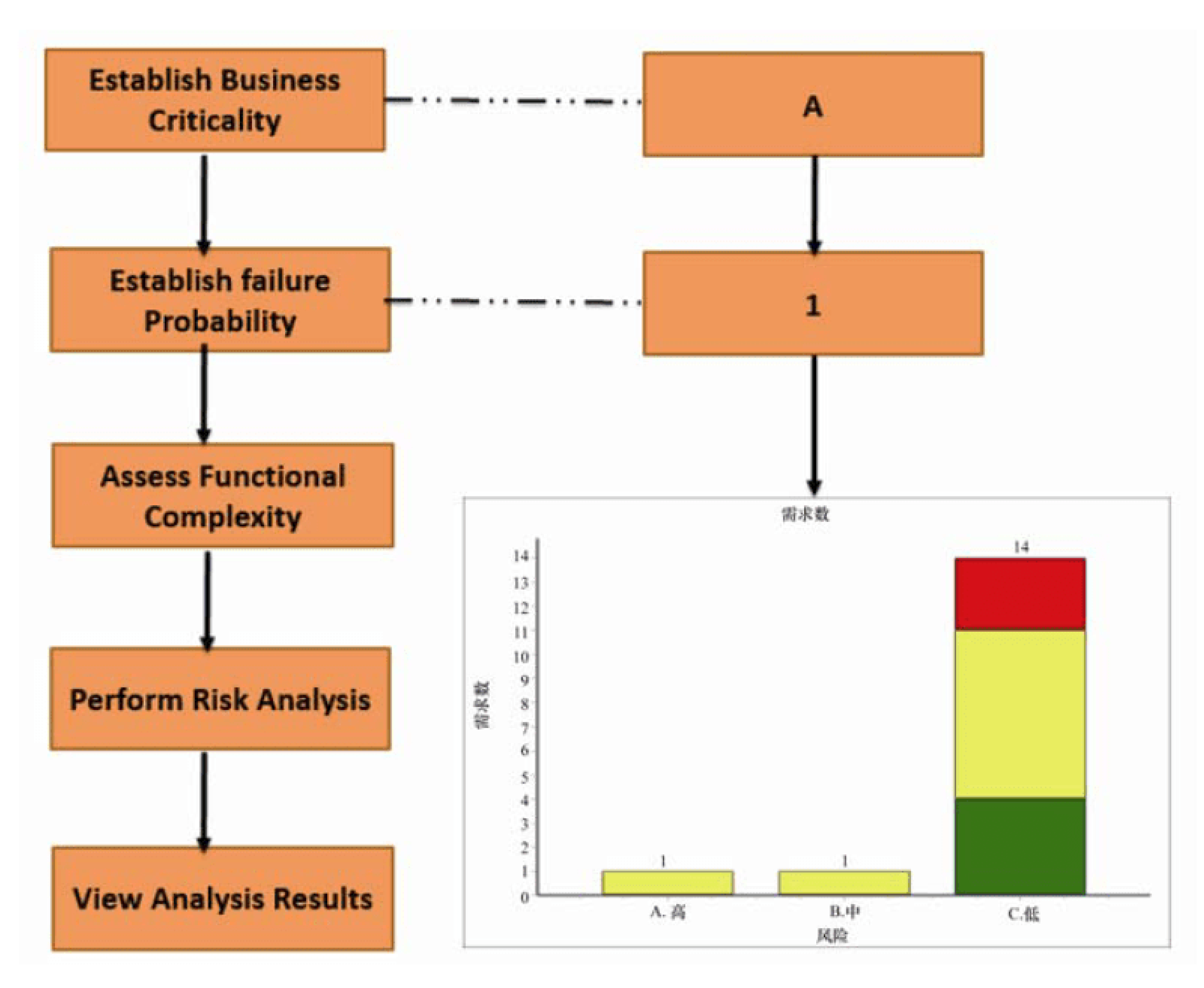


图 17-11　基于风险的质量管理流程

1. 创建业务紧急程度

确定需求对业务的紧急程度，而无需度量实现需求的难度。例如，网上银行需要用户必须可以一天 24 小时能够使用它。如果这个应用程序停止工作，它将给业务带来消极影响。因此，网上银行运用的可用性对于业务具有一个较高的紧急度。

2. 创建失败几率

确定需求的失败几率可以提前制定减缓策略。例如，如果使用网上银行功能具有较高的失败几率，可以在有大量用户同时使用网上银行时为超负荷访问部署备份服务器，这可以确保业务避免不良影响。

3. 评估功能复杂度

确定需求实现的复杂度。例如，在需求实现过程中，若应用程序会发生重大改变，可以让其和其他系统进行信息沟通，这样的需求应当设定为较高的复杂度。

4. 执行风险分析

基于需求的业务紧急程度和失败几率，为需求分配测试时间。基于测试时间，ALM 计算需求的测试级别，并运用测试级别确定需求是否进行完全测试或部分测试。

5. 查看分析结果

完成风险分析之后，应该运用图表查看分析结果。

17.7　分析和评估风险

基于需求的类型，ALM 能实现相关的风险分析或评估(如图 17-12 和图 17-13 所示)。例如，Folder 类型的需求在需求树中具有一个高的级别并拥有子需求。评估所有与子需求相关的风险，并在这些评估结果的基础上分析父需求的风险。父需求是分析需求，子需求是评估需求。

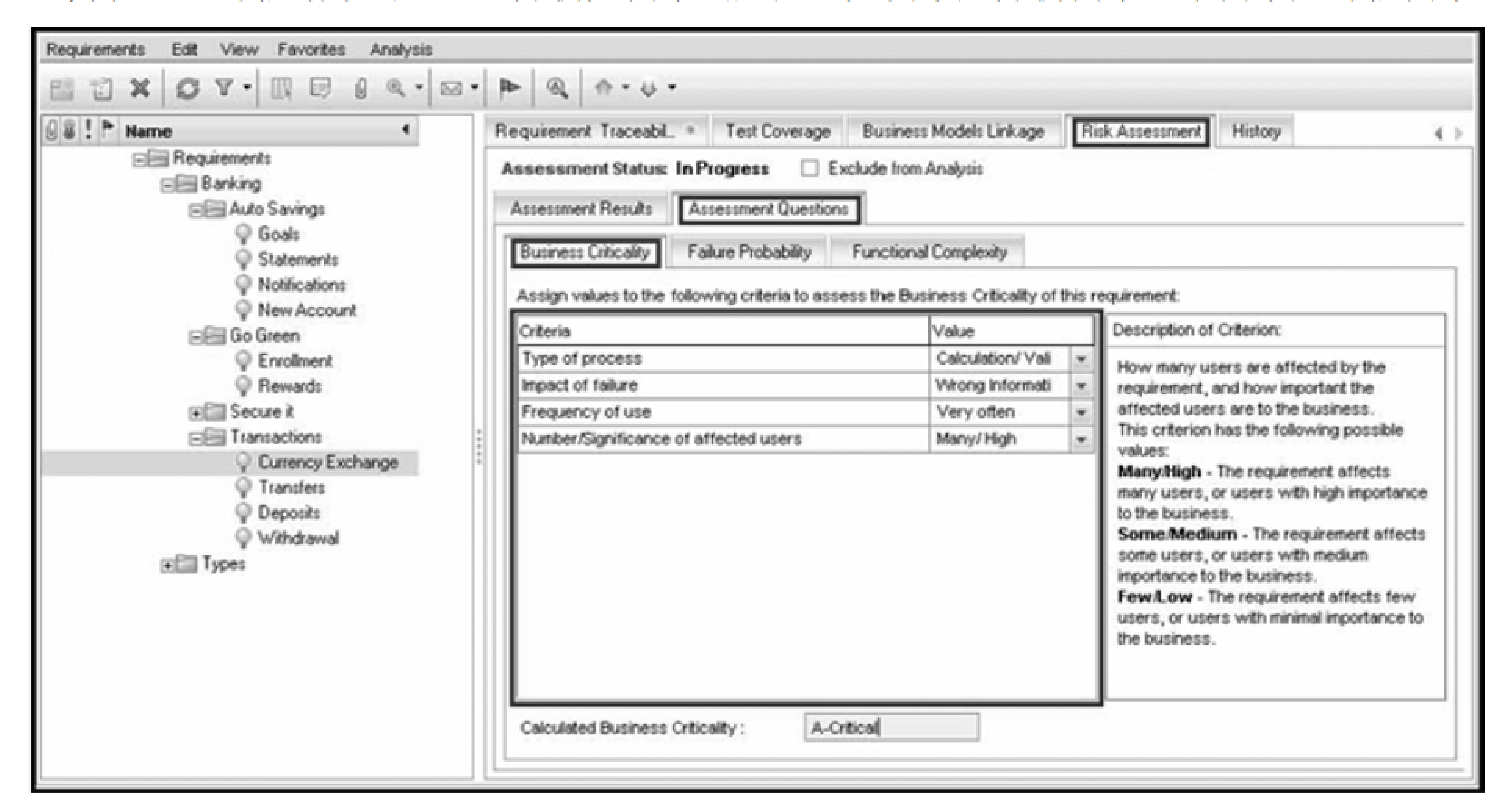


图 17-12　分析与评估风险的操作界面

1. Assessment Results 标签：这个区域显示了评估需求的风险值和功能复杂度。

2. Assessment Questions 标签：这个标签使我们通过直接给需求赋值或给一系列的条件赋值来确定业务紧急度、失败几率和功能复杂性。

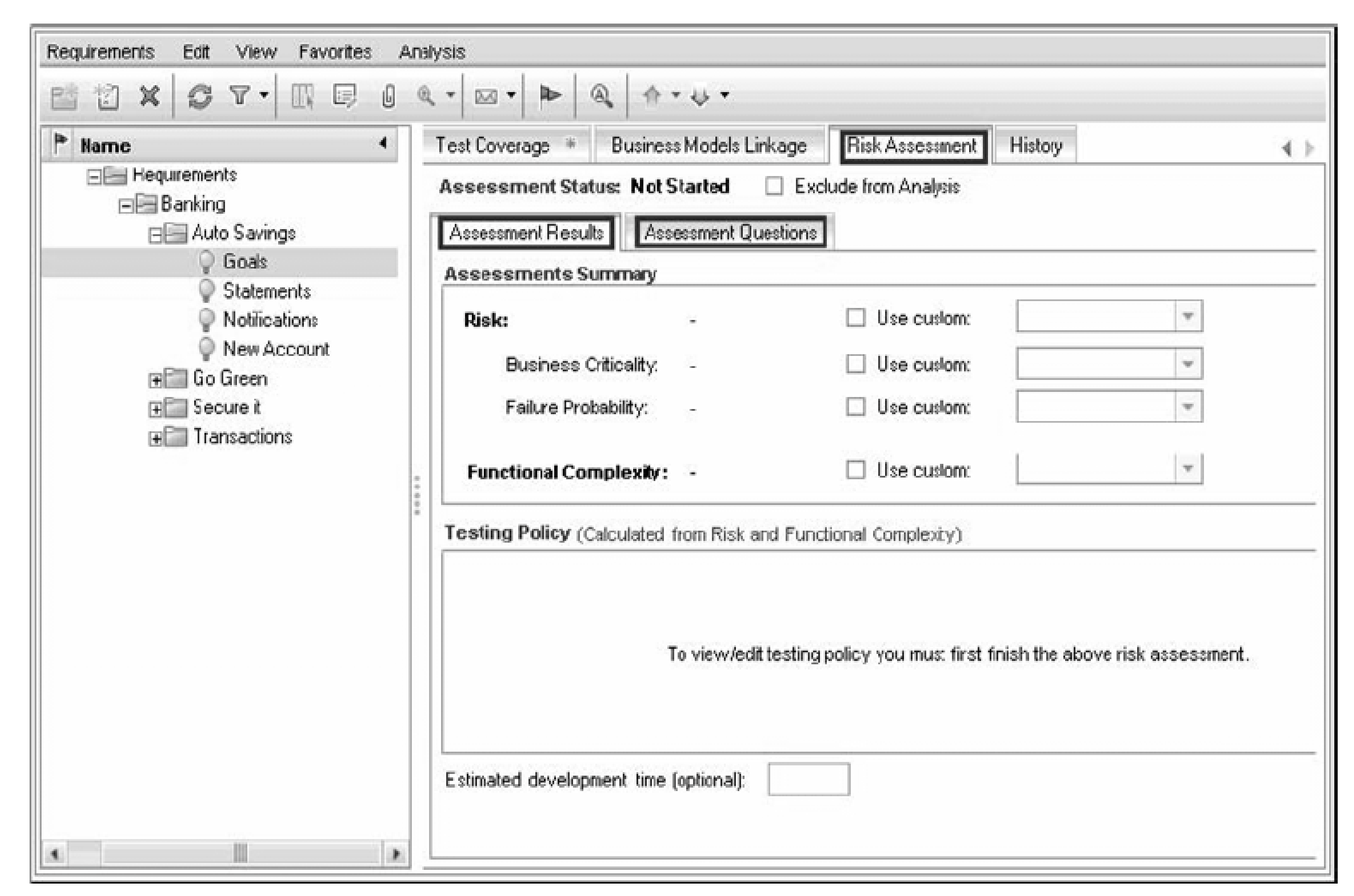


图 17-13　分析与评估风险的结果

17.8　建立业务紧急度

17.8.1　为需求创建业务紧急度

(1) 在 Requirement Details 视图中，从需求树中选择一项需求并在右侧窗口中单击 Risk Assessment 选项卡，在右边窗口中，单击 Assessment Questions 选项卡。

(2) 单击 Business Criticality 选项卡。Business Criticality 页面显示了一系列用于决定被选需求的业务紧急度的准则。

(3) 为标准赋值。从 Criterion 列中选择一个标准名称，被选标准的描述信息出现在 Description of Criterion 区域内。

(4) 从 Value 列中选择一个准则的值。

(5) 为每项标准赋值之后，ALM 通过计算需求的业务紧急度并在 Calculated Business Criticality 范围内显示结果。

17.8.2　建立失败几率

当为每项需求定义风险类型之后，需要确定每项需求所需的测试时间(如图 17-14 所示)。测试所需时间依赖于需求的失败几率。通常测试一项具有高失败几率的需求需要更多的时间，因为这项需求的实现中可能存在缺陷。

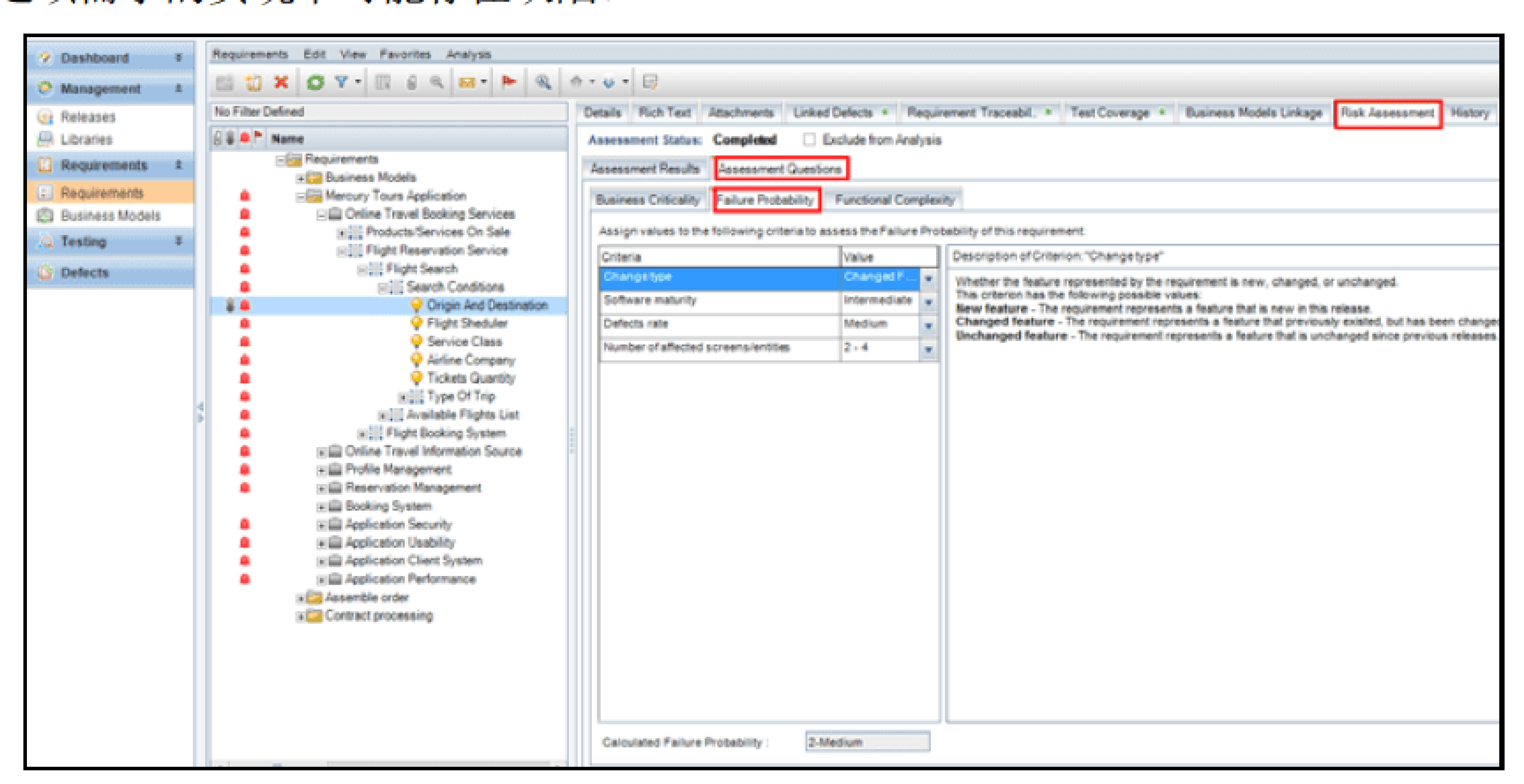


图 17-14　失败几率设定界面

17.8.3　建立需求失败几率

(1) 在右侧面板中，单击 Failure Probability 选项卡。Failure Probability 页面显示了一系列用于确定被选需求的业务紧急度的准则。

(2) 为准则赋值，从 Criteria 中选择一个准则名称。被选准则的描述信息出现在 Description of Criterion 区域内。

(3) 从 Value 列中选择一个准则的值。

(4) 在每项准则赋值之后，ALM 通过计算需求的业务紧急度并在 Calculated Failure Probability 范围内显示结果。

17.8.4　建立功能复杂度

可以指定或计算一项评估需求的功能复杂性范畴。需求的功能复杂性象征着需求实现的复杂度。它有三个可能的值：高、中、低。例如，如果一项需求的实现涉及应用程序重大变更，而它需要能和其他系统进行通信，那么它可能有一个高的复杂度，因此应该赋予高级别功能复杂度。相反，一个不涉及重大变更的需求不会与一些风险关联，因此可能赋予低级别的功能复杂度。

确定需求的功能复杂性范畴(如图 17-15 所示)：

(1) 在右侧窗口中，单击 Functional Complexity 选项卡，Functional Complexity 选项卡显示了一系列用于确定功能复杂性的准则。

(2) 为每项准则赋值。要为每项准则赋值，在 Criteria 列，单击准则的名字并从 Value 列中选择一个值。可以在 Description of Criterion 复选框中查看准则的描述信心。为每项准则赋值之后，功能复杂性将根据为准则指定的值进行更新。

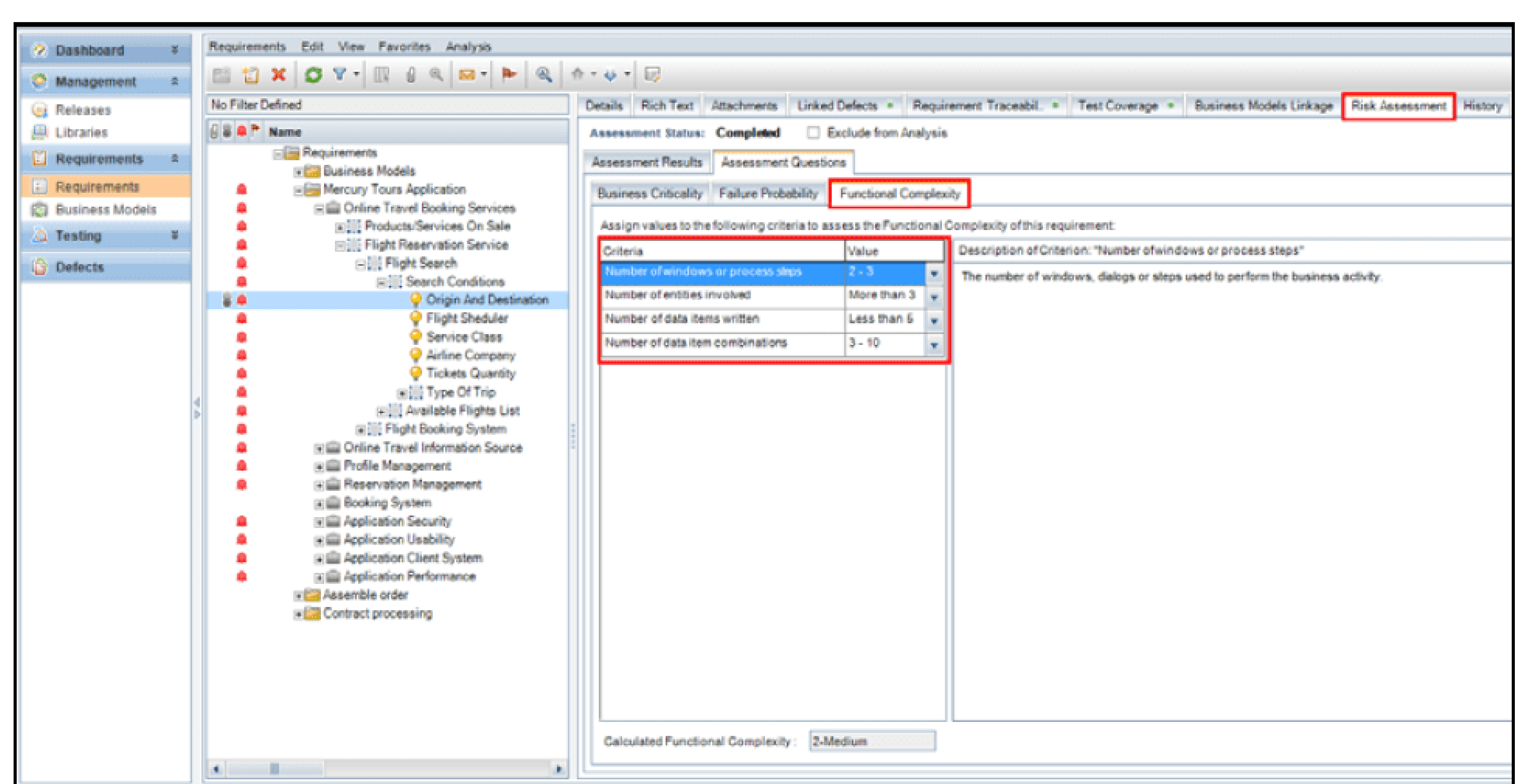


图 17-15　功能复杂度设定界面

17.9　执行风险分析

为评估需求创建失败几率后，通过为分析需求分配测试时间来执行风险分析(如图 17-16 所示)。分析可以在未指定所分配给一系列需求的总测试时间的情况下执行，但是提供测试时间将会进行更好的分析。发布的早期阶段，会根据一系列普通的准则估算测试时间以建立基

线和测量测试结果。

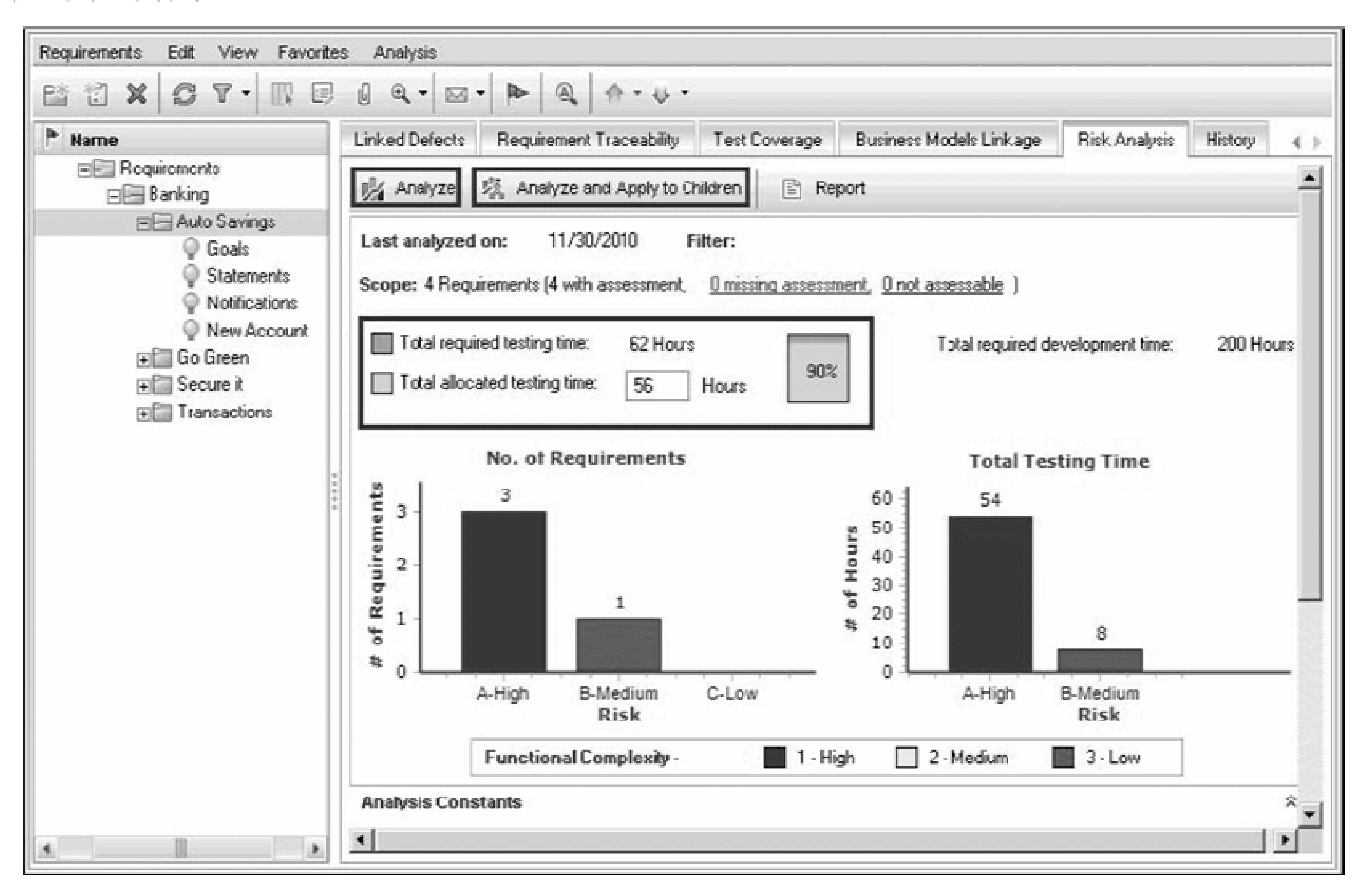


图 17-16　风险分析执行结果

页面是新的还是已存在的，是否为数据库字段添加计算值或从用户数据中获取，这类问题的答案以非常普通的测试数据阐明。例如，这个版本中所有新的页面都要分配 80 小时进行测试，然而，改变已存在的页面可能只需要 40 小时。数据库的更改需要用 60 小时建立新的字段，但是从已存在的字段中直接获取只需要 20 小时。

然后，当需求趋于稳定，这些普通的配置也可能描述地更准确。同时，当项目经理意识到测试一项特别的需求有不确定性而且意义很大的时候，时间可以在不同需求之间进行调整，因此需要更多的测试时间。分析可以帮助项目保持跟踪。

17.9.1　执行风险分析

(1) 在 Requirement Details 视图中，从需求树中选择需要分析的需求。这个需求是一个文件夹或字段组。

(2) 在右侧面板中，单击 Assessments Results 选项卡。

(3) 在 Total allocated testing time 区域内，键入测试需求所需的可用时间和子需求。

(4) 单击 Analyze and Apply to Children。ALM 基于评估需求的风险种类与分析需求的测试级别和测试时间，计算每个评估需求的测试时间和测试级别。

17.9.2　查看分析结果

风险分析计算结果以如下几种方式显示(如图 17-17 和图 17-18 所示):

1. Total Required Testing Time 区域：显示风险分析中测试所有评估需求所需要的总测试时间。

2. Total Allocated Testing Time 区域：显示开发所有评估需求的总时间，基于估计的每一

项评估需求所需的开发时间。

3. No.of Requirements 图：显示与每种风险类型相关的子需求的数量。

4. Total Testing Time 图：显示测试所有指定风险类型的需求所需的总测试结果。

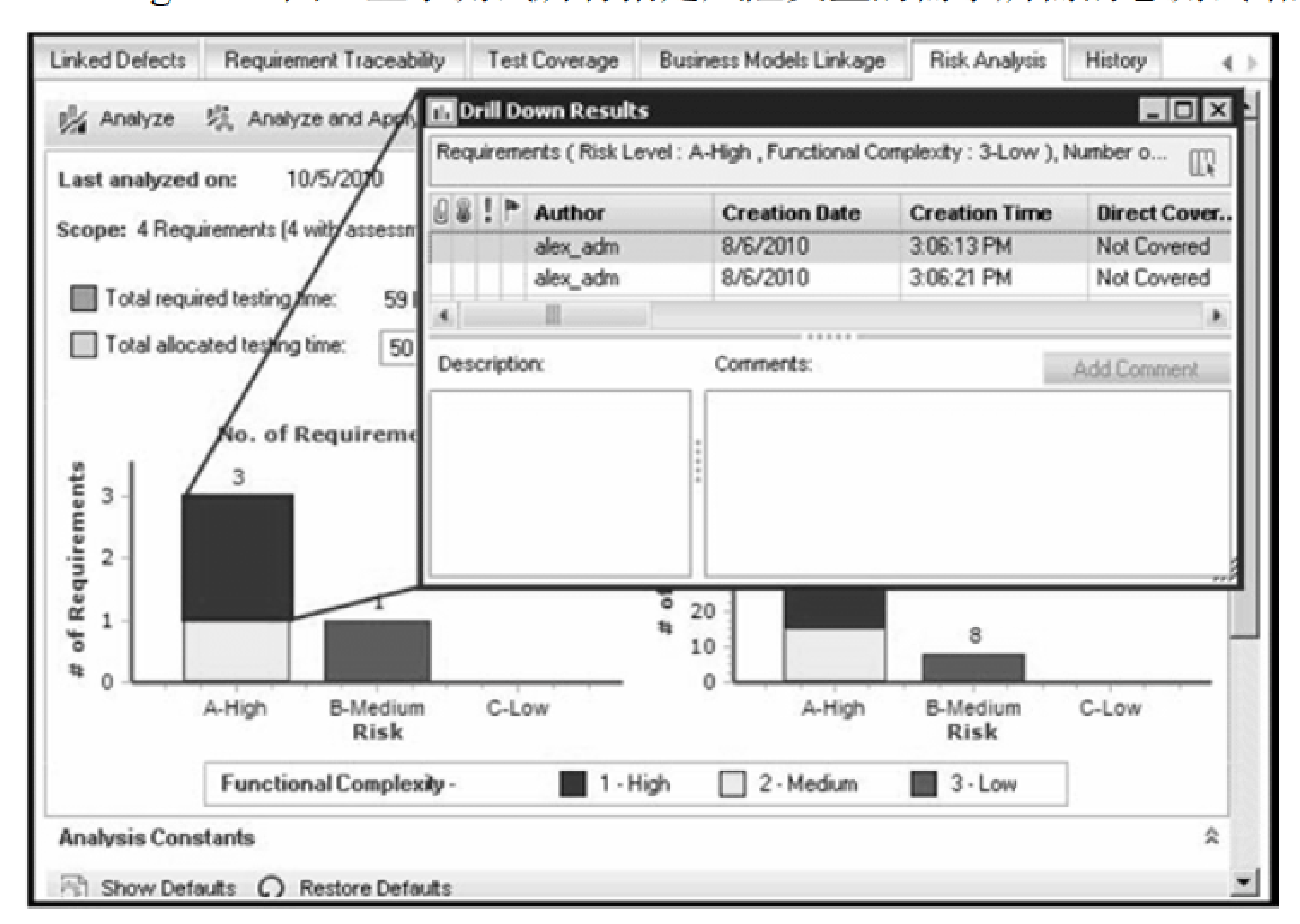


图 17-17　风险分析计算结果的查看

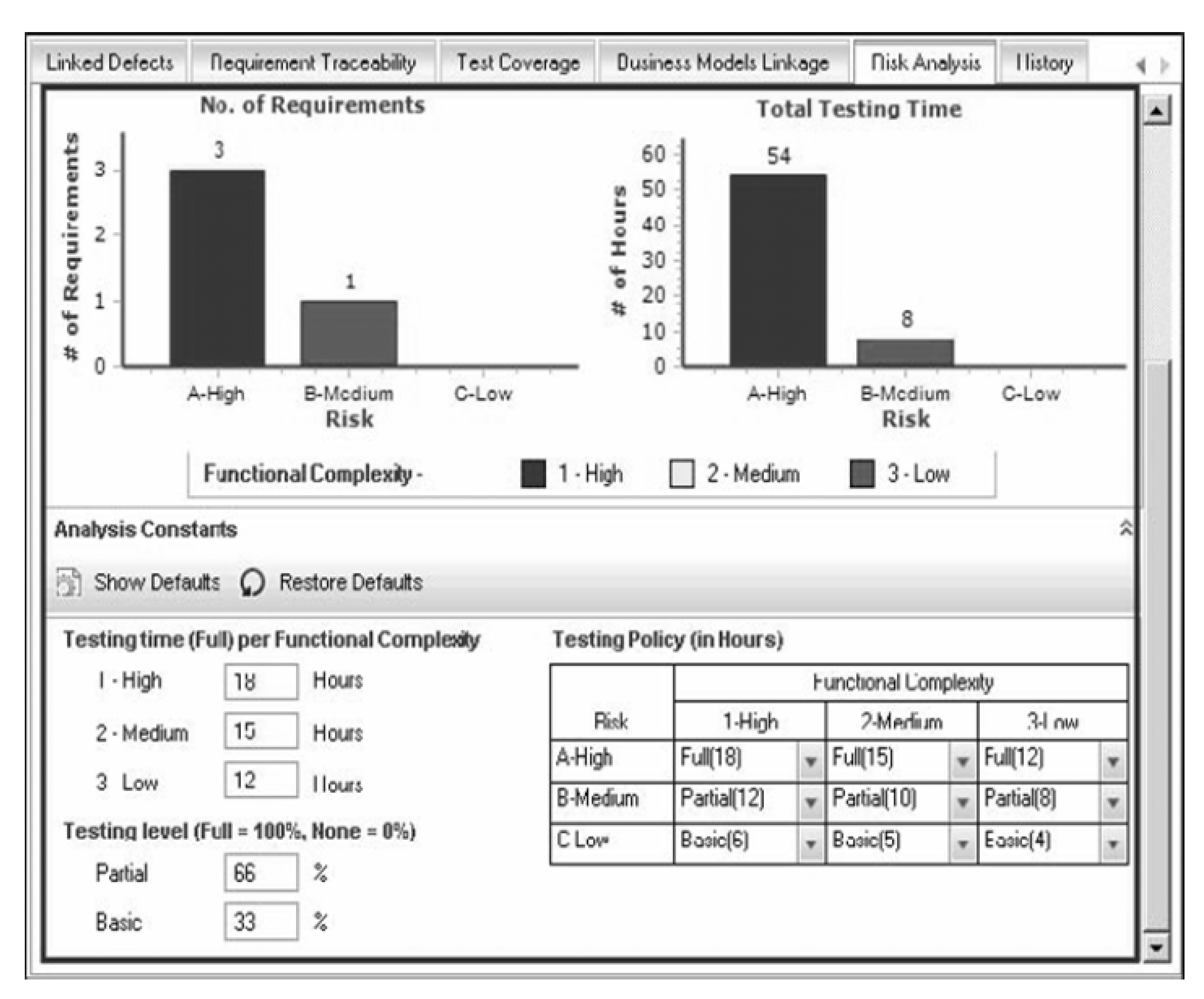


图 17-18　高风险结果详细分析

17.9.3　深入分析结果

查看每种风险类别中的需求，单击 No.of Requirements 图中的某一分段。弹出 Drill Down

Results 窗口，窗口中包含风险类别中的需求列表。可以自定义字段的顺序和外观并查看某个需求的详细信息。也可以将网格的内容以文本、表格、Word 或 HTML 文档形式导出。

要显示不包含在分析中的需求，单击 Excluded 链接。弹出 Drill Down Results 窗口，窗口中包含一个不包含在分析结果里的需求列表。

查看并深入分析结果之后，比较计算出的测试总时间和可用资源的量(如图 17-19 所示)。如果可用资源的数量不满足测试需求，那么就应该减少评估需求的测试级别或减少每项测试级别的测试时间。在减少测试级别和测试时间后再次执行风险分析。

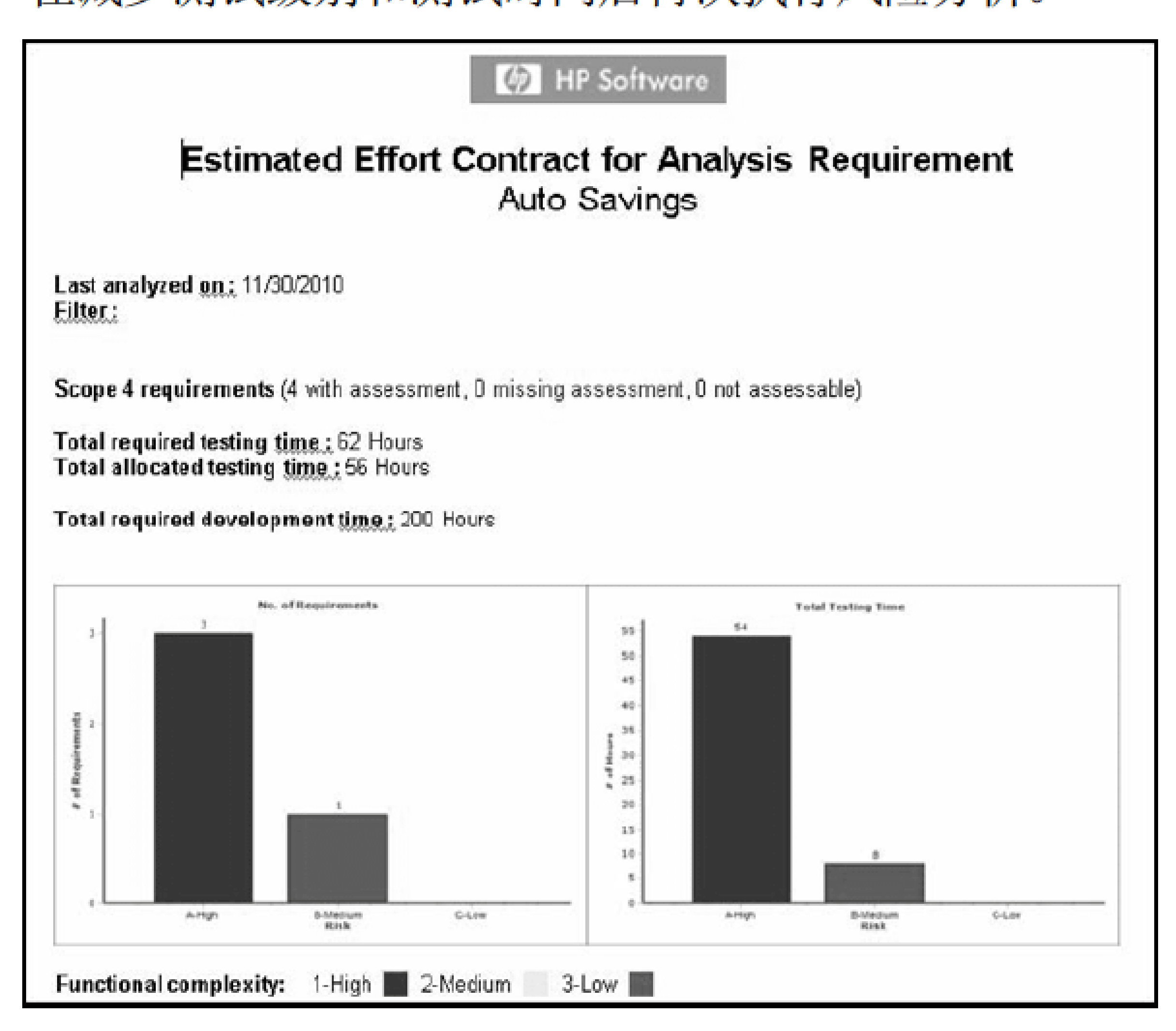


图 17-19　基于需求的测试工作量分配图

17.9.4　生成风险报告

完成需求的测试策略之后，可为分析需求生成一个描述测试策略的风险报告(如图 17-20 所示)。

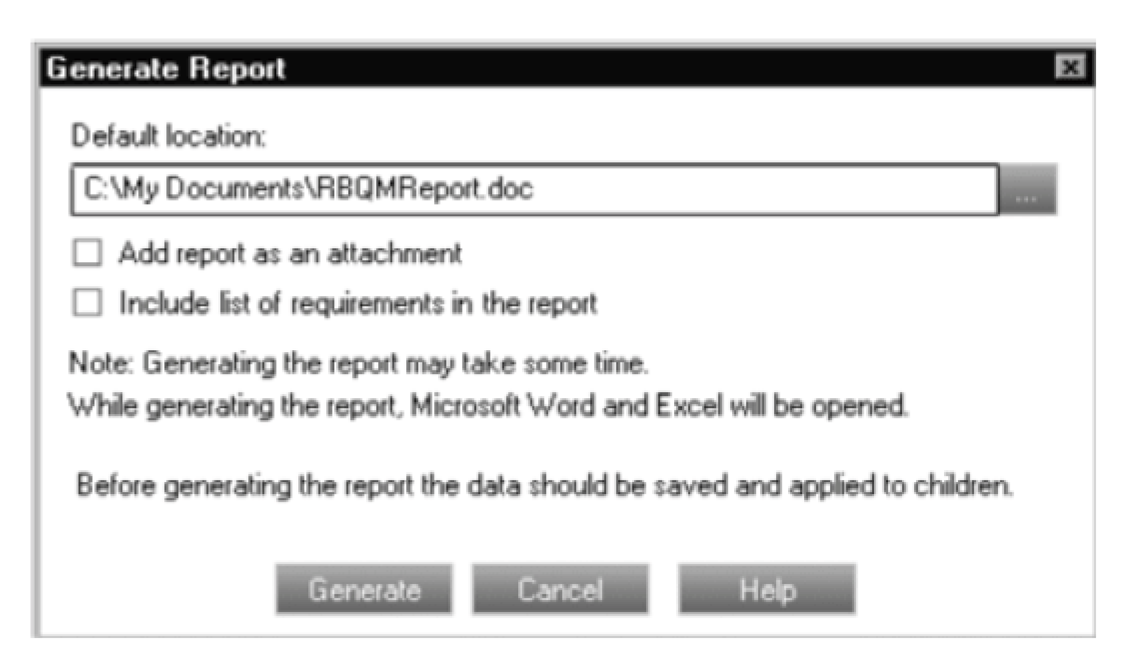


图 17-20　风险报告生成

(1) 在 Risk 页面上，单击 Report 按钮。弹出 Generate Report 对话框。

(2) 在 Generate Report 对话框中，在 Default Location 区域里，单击浏览按钮，从 Save As 对话框中选择保存位置。

(3) 要以附件方式将报告添加到分析需求中，选择 Add report as an attachment 选择框。

(4) 要在风险分析中包括评估需求列表，选择 Include list of requirements in the report 选择框。

(5) 单击 Generate。ALM 在指定位置生成并保存了报告。

17.10　测试范围

在需求树中定义需求，然后再分派到版本树上的某一版本或周期上。在计划阶段，在这些分配的需求的基础上建立测试计划树。保持跟踪分配的需求与测试之间的关系(如图 17-21 所示)。在测试计划模型中，可通过选择要链接到测试的需求来创建需求范围。或者，也可以在需求模型中，通过选择要链接到需求的测试来创建测试范围。一个测试可涵盖多个需求，而一个需求也可被多个测试涵盖。

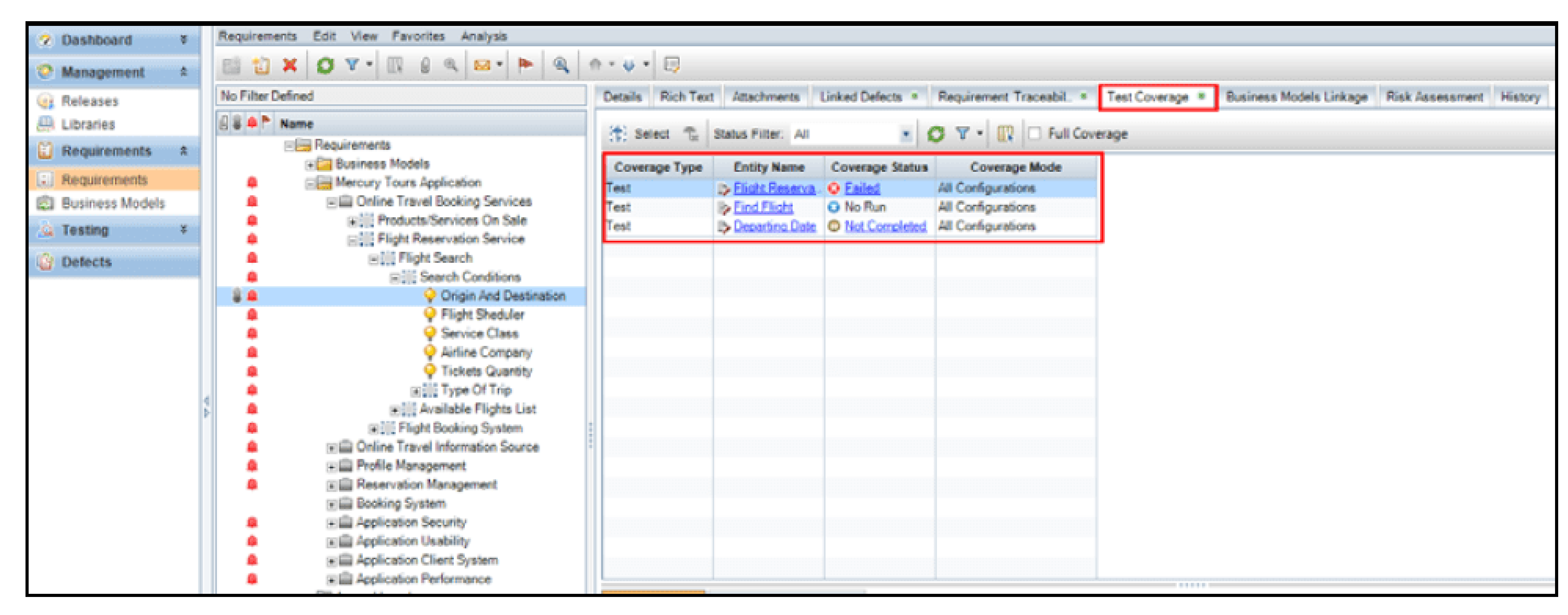


图 17-21　需求与测试的涵盖关系

17.11　关联缺陷

缺陷可以与以下实体相关联：需求、测试、测试集、测试实例、运行项、运行步骤以及其他缺陷。缺陷关联是非常有用的，例如在专门对已知缺陷新建测试时。通过创建关联，可以确定是否应该基于缺陷的状态运行测试。

缺陷可以直接或间接地与一个实体关联起来(如图 17-22 所示)。当添加一个与实体关联的缺陷时，ALM 添加一个对这个实体的直接关联和其他实体的间接关联。相反，当手动运行测试添加缺陷时，ALM 会自动创建测试运行和新缺陷之间的关联。

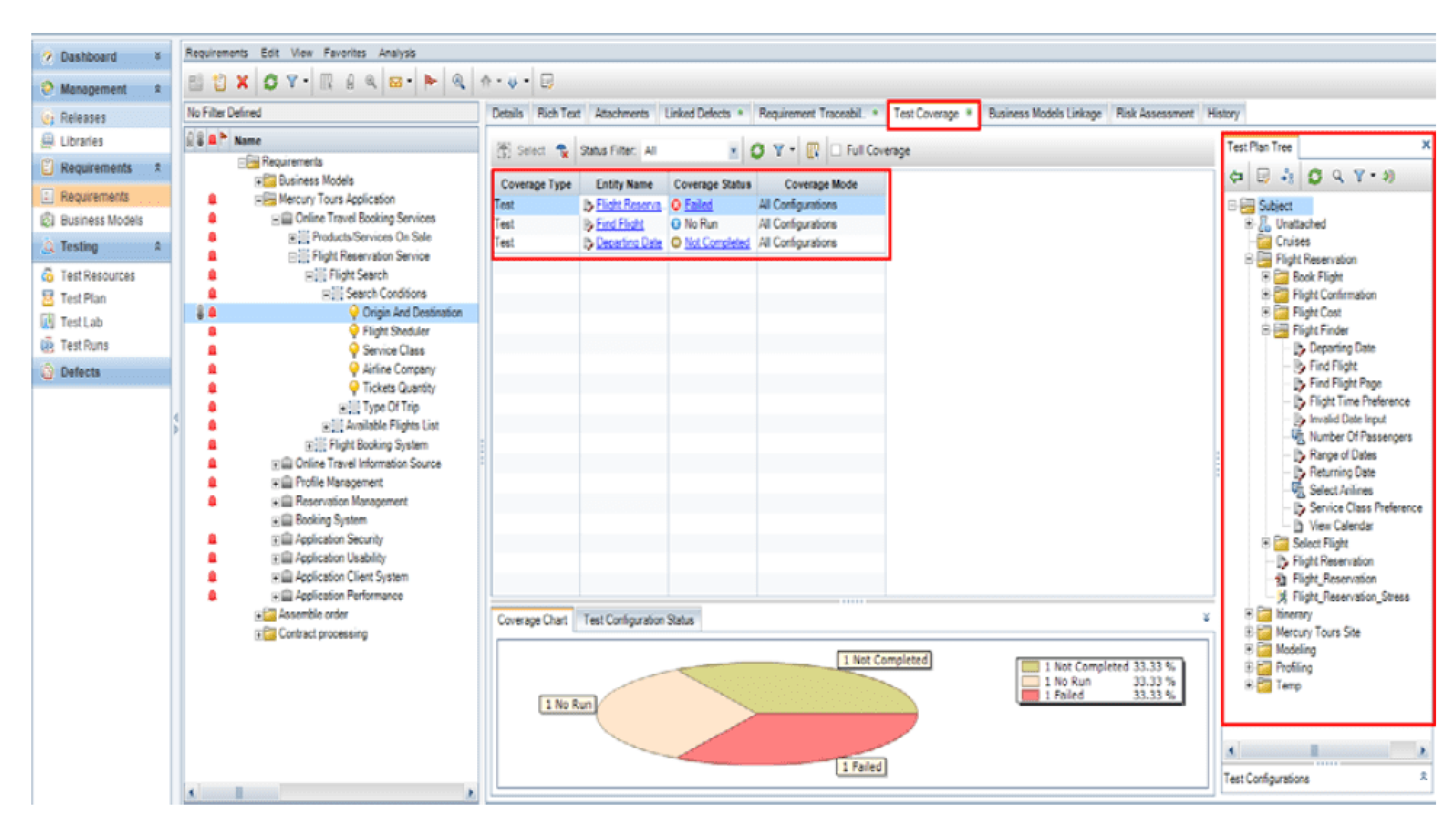


图 17-22　缺陷关联操作界面

17.12　邮件发送需求

给你所在的 ALM 项目中的其他用户发送需求相关的邮件(如图 17-23 所示)。这样可以通知他们关于需求的状态。邮件中包含一个链接，邮件接收者单击这个链接可以直接链接到相应的需求。

注意：默认情况下，ALM 以 HTML 格式发送邮件。为了以纯文本形式发送邮件，编辑 Site Configuration 选项卡中的 MAIL_FORMAT 参数。

按照下列步骤使用发送需求邮件：

(1) 选择一个或多个需求，并单击 Send by E-mail 按钮。弹出 Send E-mail 对话框(如图 17-24 所示)。

小技巧：可以自动给指定用户类型发送邮件。这个可以是任何用户姓名值所在列，包括用户定义范围。单击 Send by E-mail 所示箭头并选择一个选项。例如，选择 Send by E-mail to Author 来给编写此需求的人发送邮件。

(2) 添加可用的邮件地址或用户名。或者单击 To 按钮和 CC 按钮选择用户，弹出 Select Recipients 对话框。可以通过用户组并从列表或者用户组中选择用户来对用户列表进行分类，对用户进行搜索，对用户进行分组。

(3) 在 Subject 框中，填写邮件主题。

(4) 确定是否要包含附件、历史记录、测试范围或者是需求的跟踪需求。如果包含需求的附件，任何需求的富文本将作为单独的附件。

(5) 单击 Send 发送邮件。

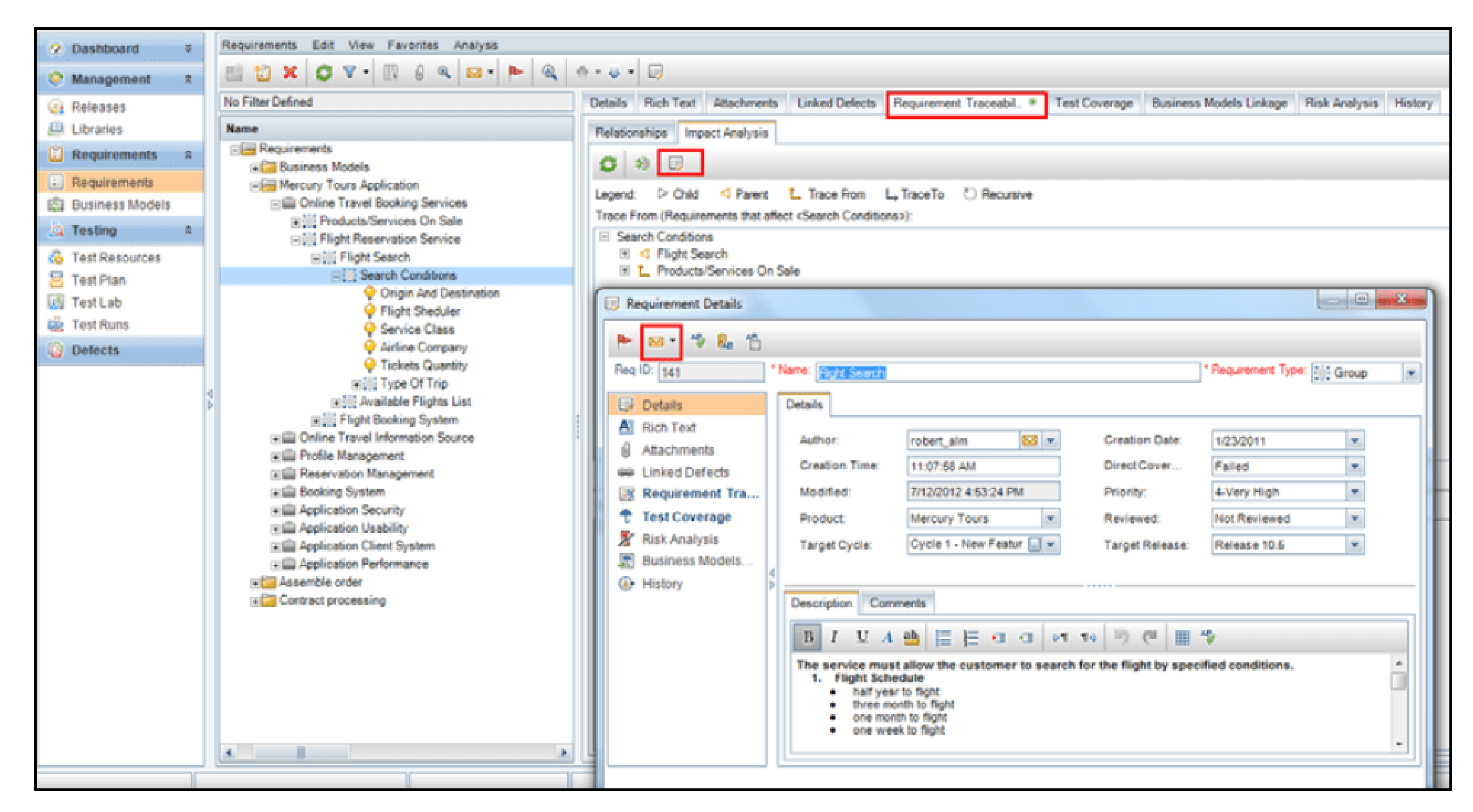


图 17-23　发送需求相关的邮件

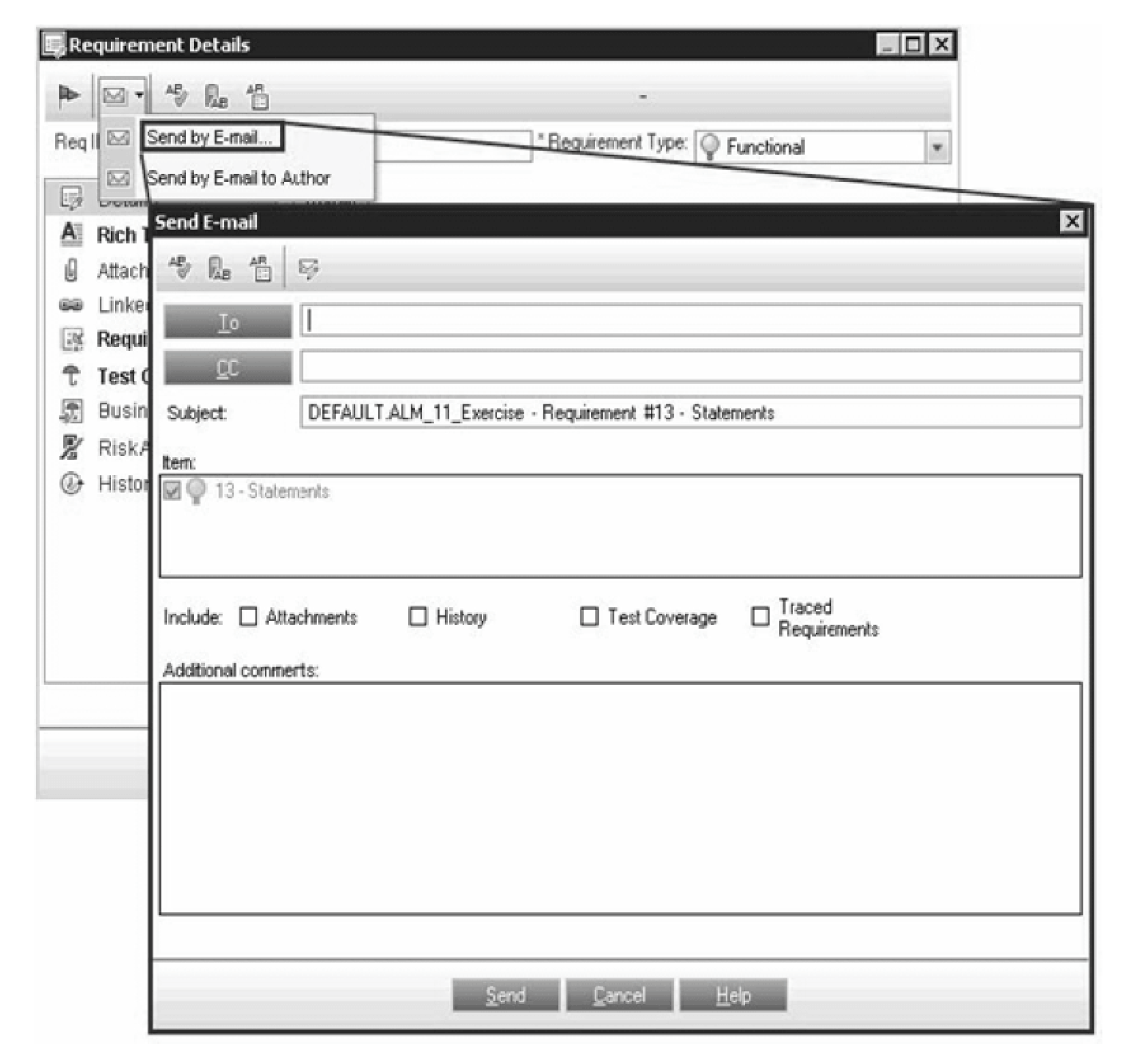


图 17-24　邮件发送界面

习题与思考题

1. 在需求之间添加跟踪链的目的是什么？
2. 在风险分析中，哪些是计算测试需求所需的总时间的关键点？

练习：需求风险分析

在向测试发布和测试周期设定需求之后，可以在需求之间添加跟踪链以建立他们之间的依赖关系并识别与需求相关的风险。添加需求之间的跟踪链之后，可以通过为 Create Order、Update Order 和 Delete Order 需求建立业务关键性和失败可能性来计算与 Business Processes 需求文件夹相关联的风险。

在本练习中，你会执行下列任务：

第 1 部分：在需求之间添加跟踪链

第 2 部分：执行风险分析

(该练习的指导步骤请参见本章正文)

第18章　需求覆盖

18.1　简　　介

测试计划中的测试活动必须满足最初的需求。只有在需求和测试之间添加连接，才能对它们之间的关系保持跟踪。在测试计划模块中，通过选择与测试相连接的需求来确定需求覆盖范围。需求覆盖范围可以用来辅助对测试或需求变化的影响进行评估。一个测试可以覆盖多个需求。而在另一种情况下，在需求模块中，通过需求连接测试来确定测试覆盖范围。测试覆盖范围同样辅助对测试或需求变化的影响进行评估。一个需求可以被多个测试所覆盖。也可以通过测试配置覆盖需求，代替测试覆盖。测试配置是描述专门测试用例的定义组。例如，测试配置可以指定测试使用的专门的子数据集或运行环境。测试配置和需求的联合为通过不同的测试用例来确定需求覆盖范围提供了出色的执行粒度。

18.2　需求和测试覆盖范围

首先在需求树中定义需求，然后将它们分配给某个发布版本或发布树的周期。在计划阶段，基于这些需求建立测试计划树。为了保持对测试和分配的需求之间关系的跟踪，需要为它们添加连接。在测试计划模块，通过选择与测试相连接的需求来确定需求覆盖范围(如图 18-1 所示)。而在另一种情况下，在需求模块中，通过选择需求连接到测试来产生测试覆盖范围。

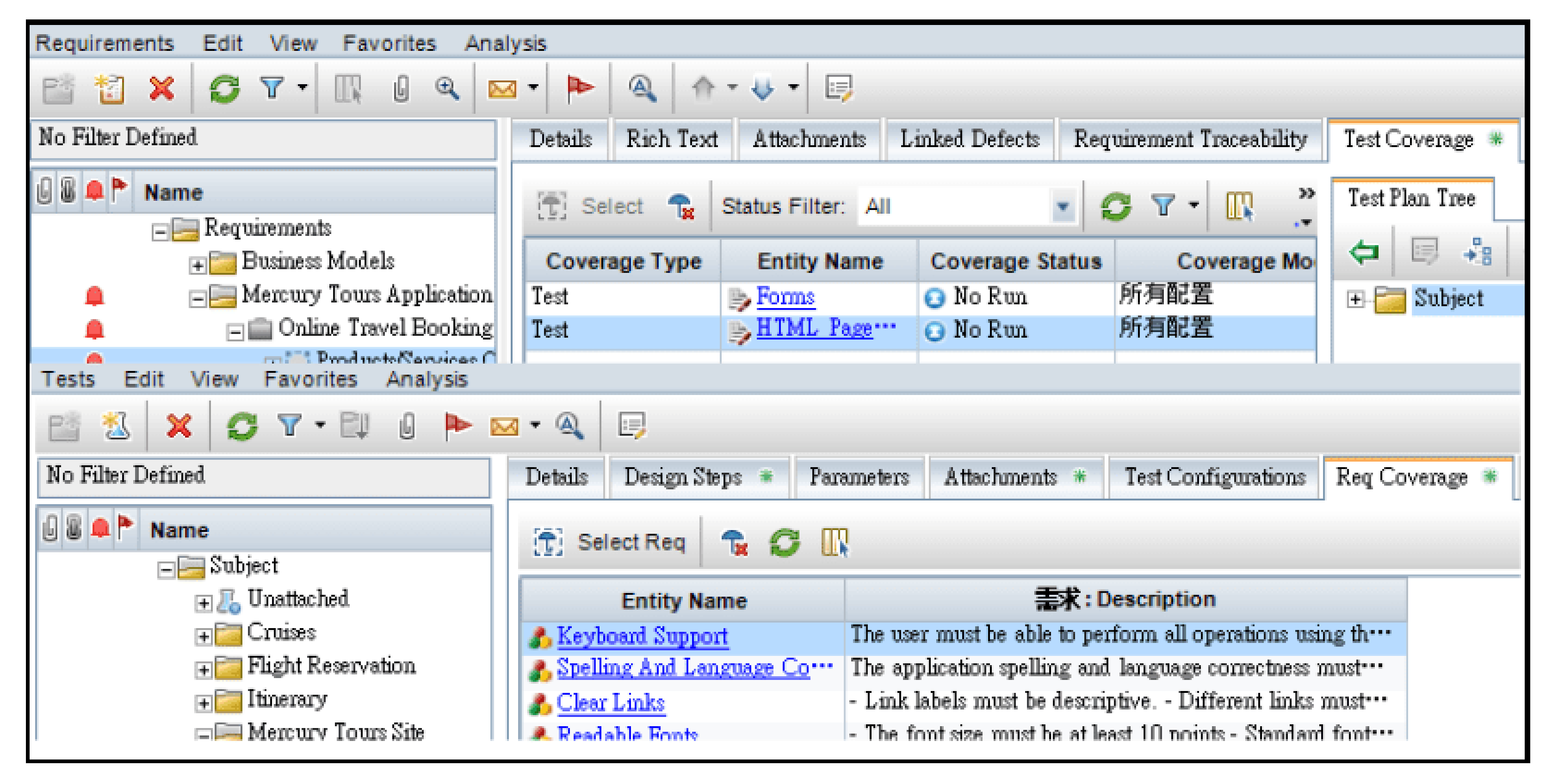


图 18-1　需求与测试计划模块

18.2.1　生成测试覆盖范围

在需求模块中，通过将测试与需求连接来生成测试范围(如图 18-2 所示)。测试覆盖范围同样辅助对测试或需求变化的影响进行评估。一个需求可以被多个测试所覆盖。

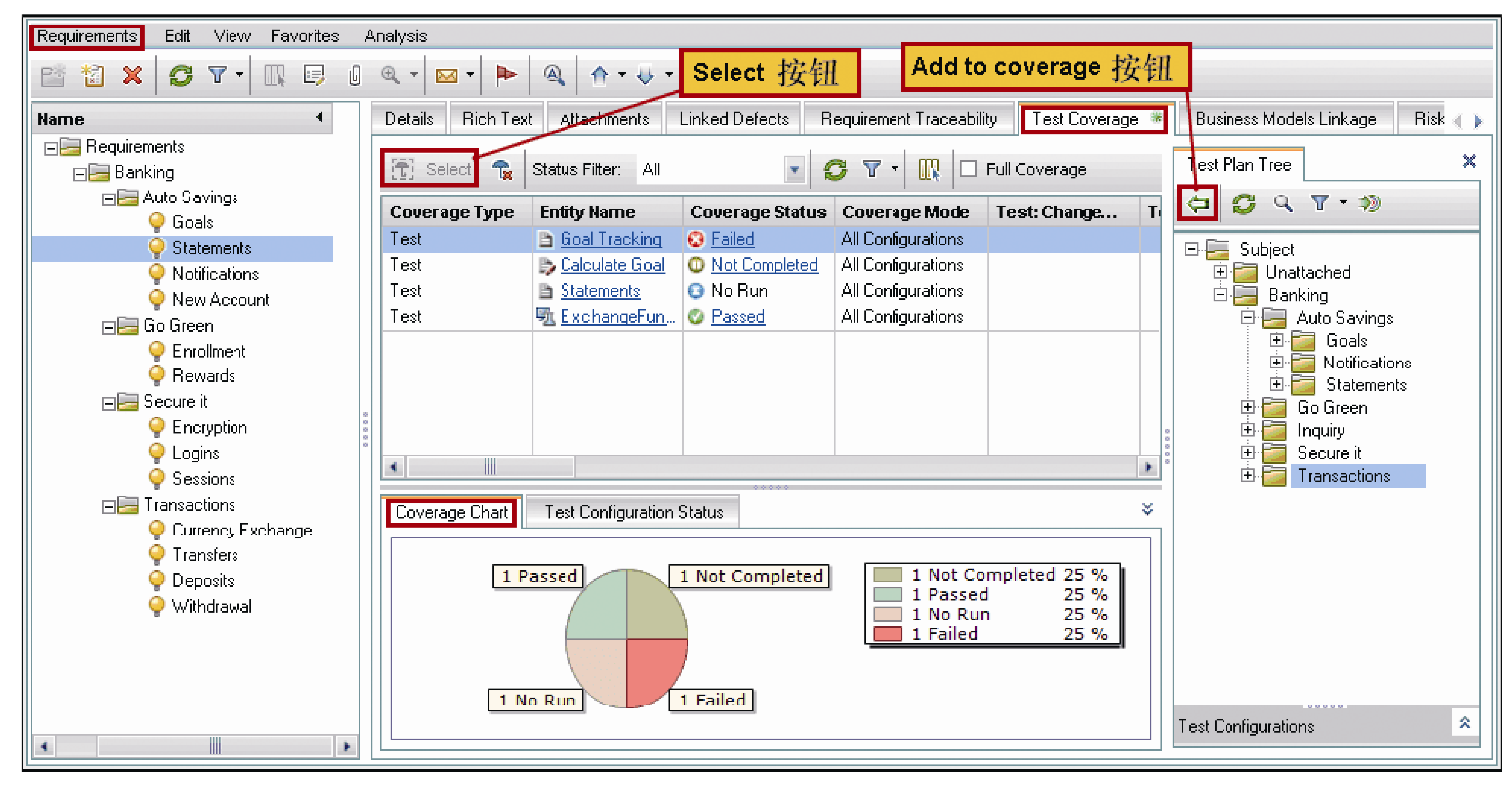


图 18-2　将测试与需求连接以生成测试范围

将测试与需求相连接的步骤如下：

(1) 在 Requirements 模块中，首先选择 View，然后选择 Requirement Details。

(2) 选择一个需求，然后单击 Test Coverage。

(3) 单击 Select 按钮，在右侧面板展开测试计划树。

(4) 选择要添加到需求覆盖范围中的测试。

(5) 单击 Add to coverage 按钮，添加选择的测试。

(6) 使用 Coverage Chart，观察覆盖范围的状态。

注意：可以将需求和测试连接到缺陷。这可以确保整个应用程序管理进程符合测试需要。如果需求发生变化，可以马上确定哪些测试和缺陷受到了影响，由谁负责。

18.2.2　生成需求覆盖范围

在测试计划模块中，通过选择与测试相连接的需求来确定需求覆盖范围，如图 18-3 所示。需求覆盖范围辅助对测试或需求变化的影响进行评估。一个测试可以覆盖多个需求。

将需求连接到测试的步骤如下：

(1) 在“测试计划”模块中，选择 View，然后选择 Test Plan Tree。

(2) 选择一个测试，然后单击 Req Coverage。

(3) 单击 Select Req 按钮，在右侧面板中展开需求树。

(4) 选择要添加到测试覆盖范围中的需求。

(5) 单击 Add to coverage 按钮，添加所选的需求。

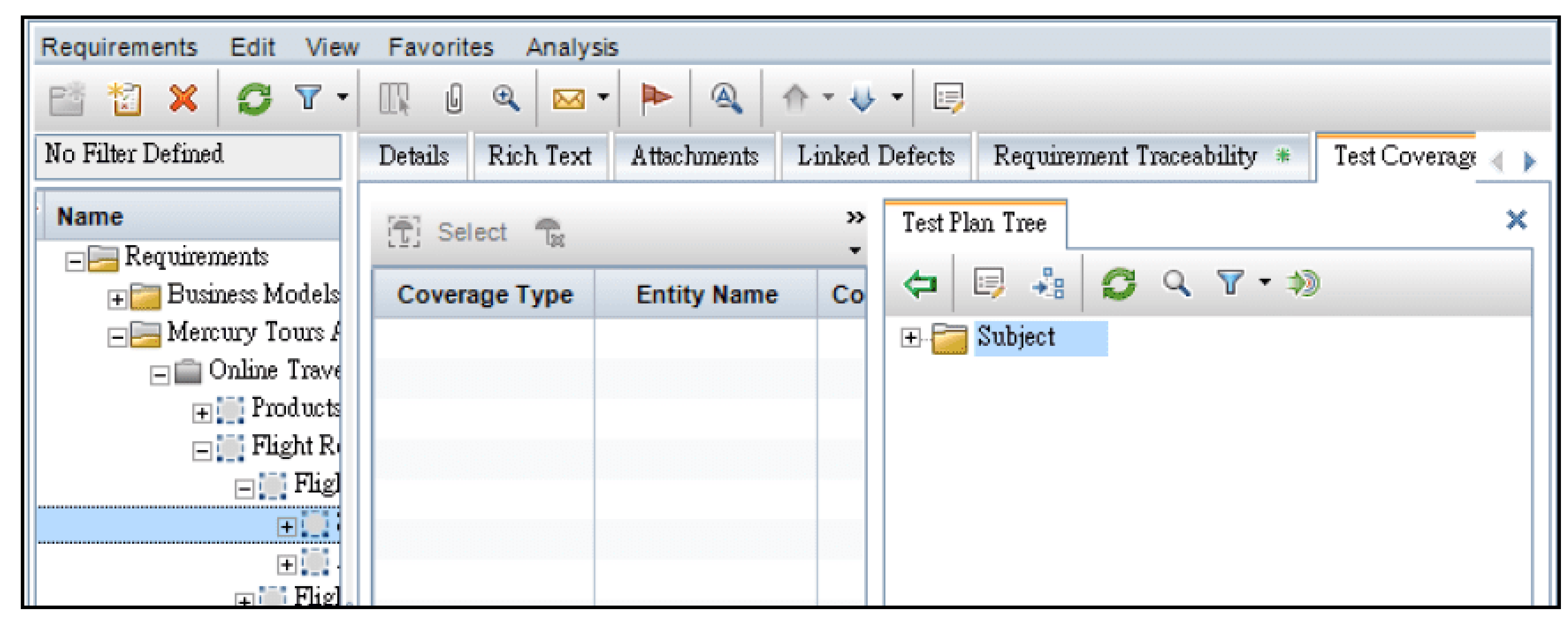


图 18-3　选择与测试相连接的需求来确定需求覆盖范围

18.2.3　测试配置覆盖范围

可以通过测试配置覆盖需求，代替测试覆盖，如图 18-4 所示。测试配置是描述专门测试用例的定义组。例如，测试配置可以指定测试使用的专门的子数据集或运行环境。测试配置和需求的联合为通过不同的测试用例来确定需求覆盖范围提供了出色的执行粒度。

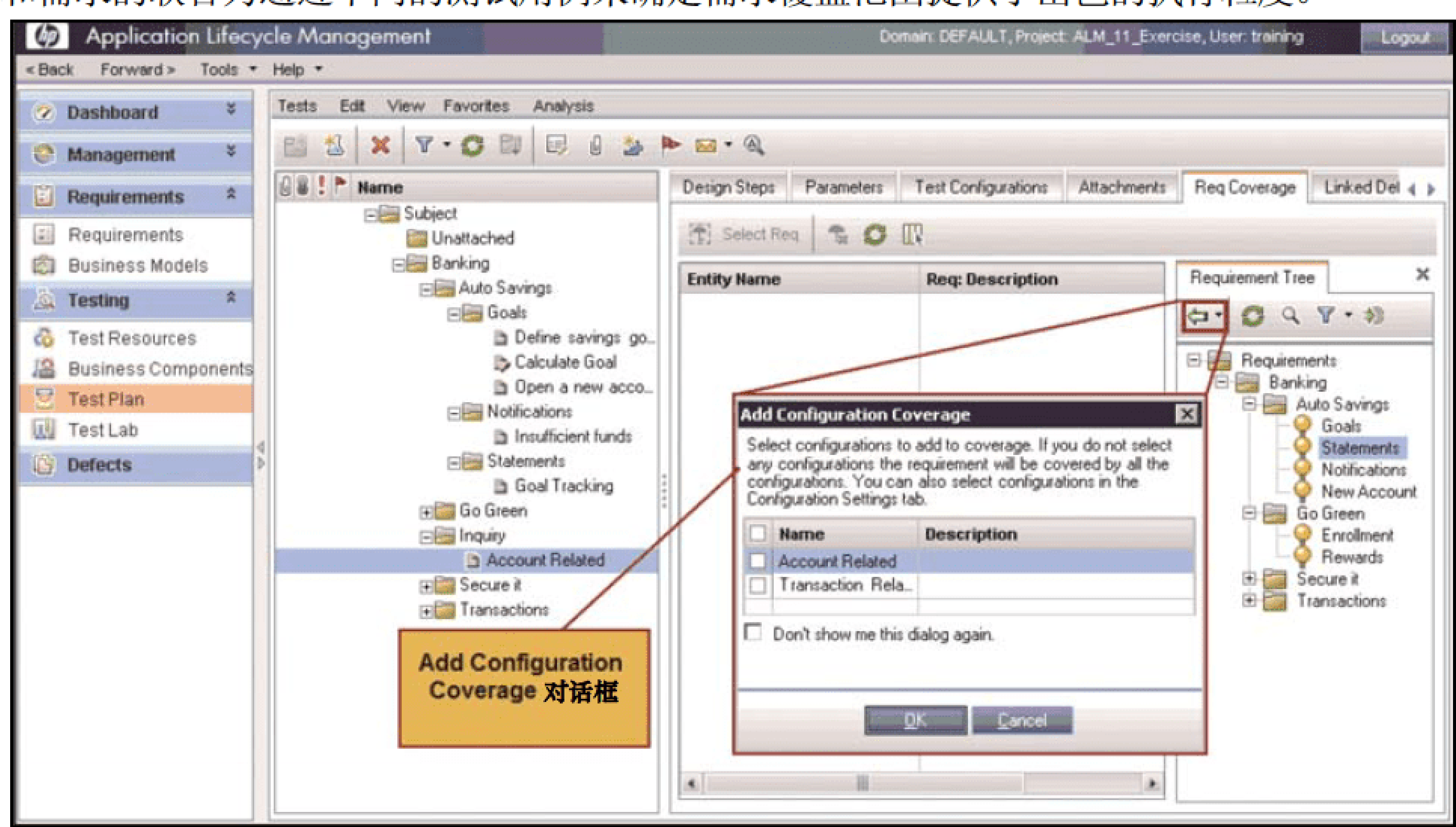


图 18-4　配置覆盖需求

通过测试配置生成需求覆盖范围的步骤如下：

(1) 在 Req Coverage 表单中，生成介于需求和测试之间的需求覆盖范围。

(2) 单击 Add to coverage 按钮。Add Configuration Coverage 对话框自动打开，显示可用的测试配置。

(3) 选择测试配置，单击 OK 按钮添加配置到覆盖范围。在 Test Configuration Settings 表单中列举了覆盖范围中的测试配置。

18.2.4　分析覆盖范围

基于测试状态，通过直接覆盖状态变化来分析覆盖范围如图 18-5 所示步骤如下：

(1) 在需求模块中，在 View 菜单中选择 Coverage Analysis。

(2) 展开 Requirements 需求树，观察子需求。

(3) 单击 Legend 链接，观察覆盖范围分析列的说明。

(4) Direct Cover Status 列根据测试状态显示了需求状态和子需求。

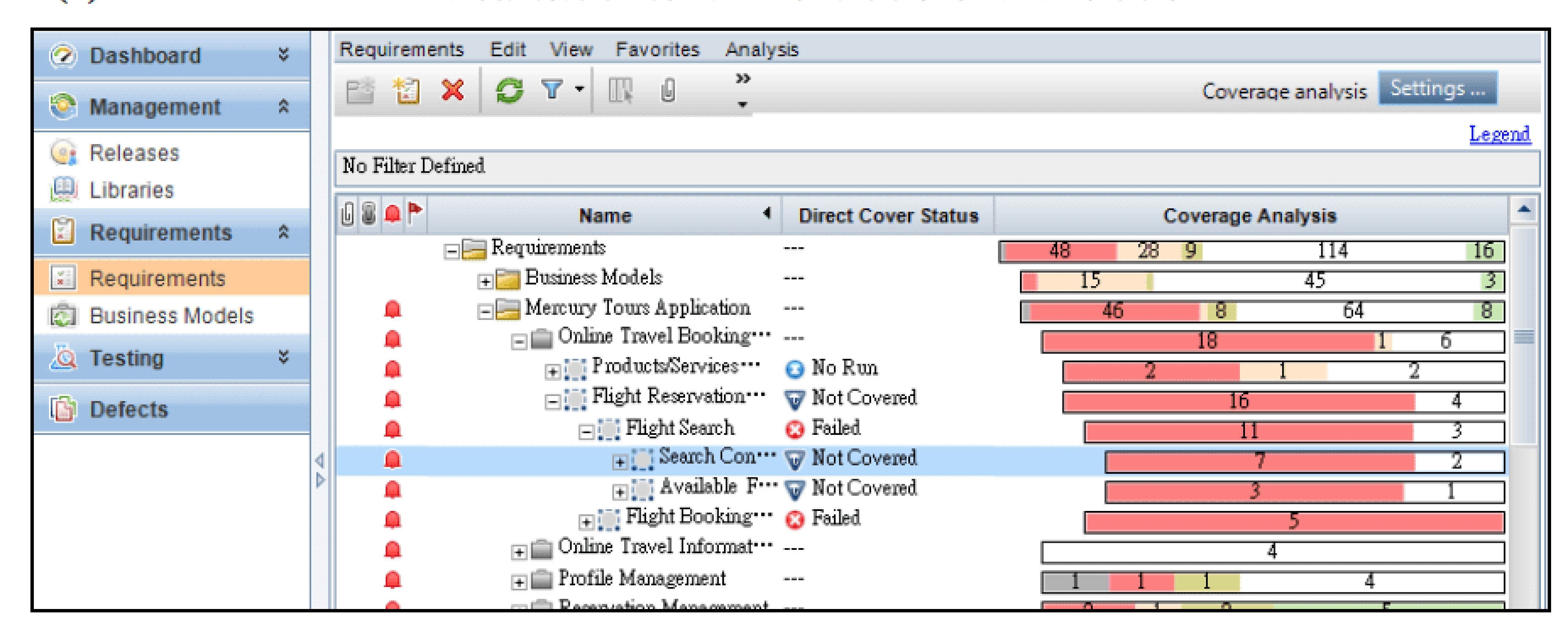


图 18-5　通过测试直接覆盖的状态变化来分析覆盖范围

18.3　追踪周期进程

追踪一个周期中的执行进程(如图 18-6 所示)，步骤如下：

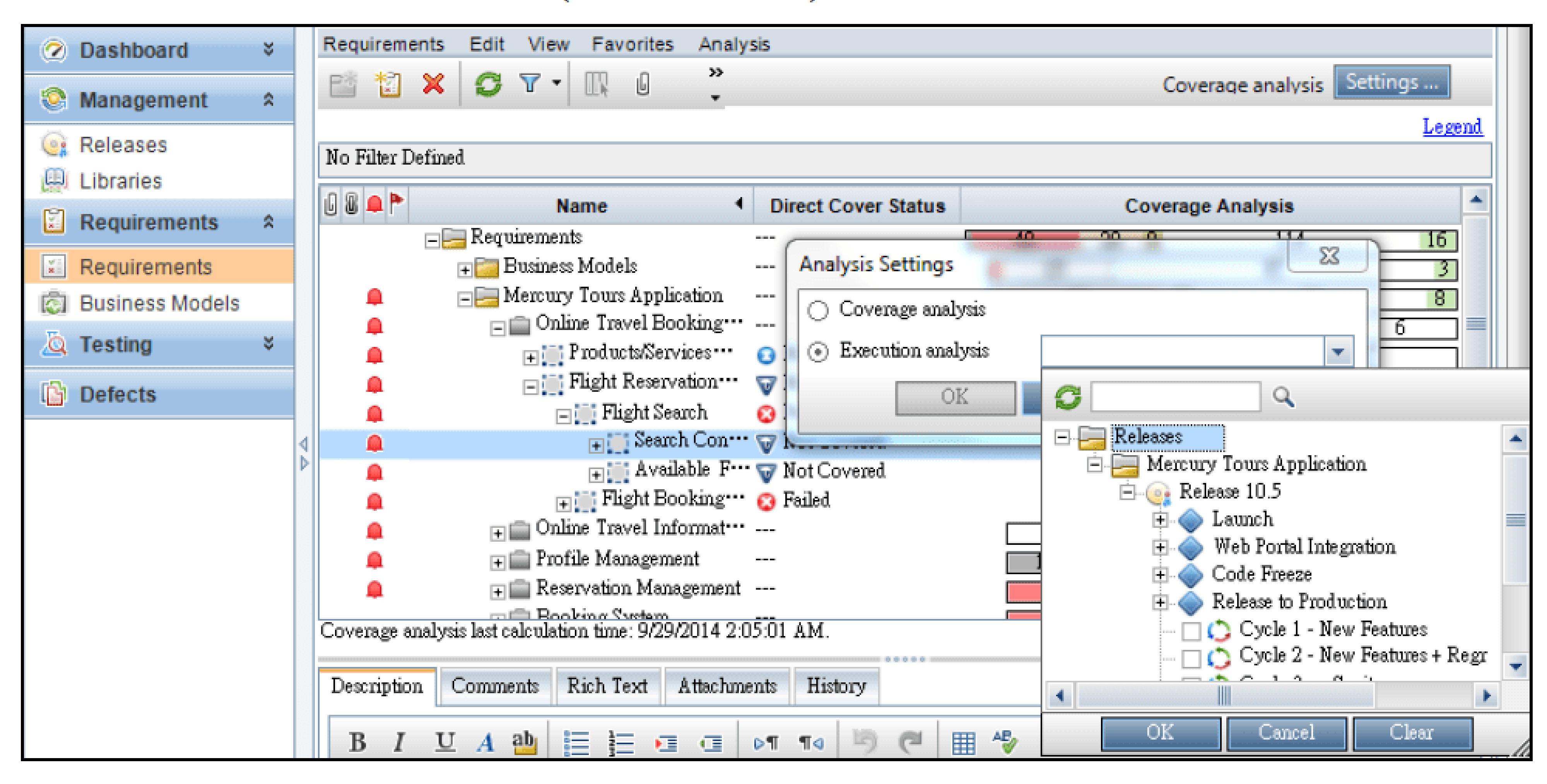


图 18-6　追踪一个周期中的执行进程

(1) 单击 Settings 按钮，打开 Analysis Settings 对话框。

(2) 选择 Execution Analysis 选项。

(3) 单击 Execution Analysis 下拉选项，Releases 树出现。

(4) 展开发布树，选择一个周期。

(5) 单击 OK 按钮，关闭这两个对话框。

通过周期观察覆盖范围(如图 18-7 所示)，步骤如下：

(1) 展开 Requirements 树。

(2) 单击 Legend 链接，查看 Coverage Analysis 的说明列。

(3) 通过选择周期，观察 Coverage Analysis 的覆盖分析列。

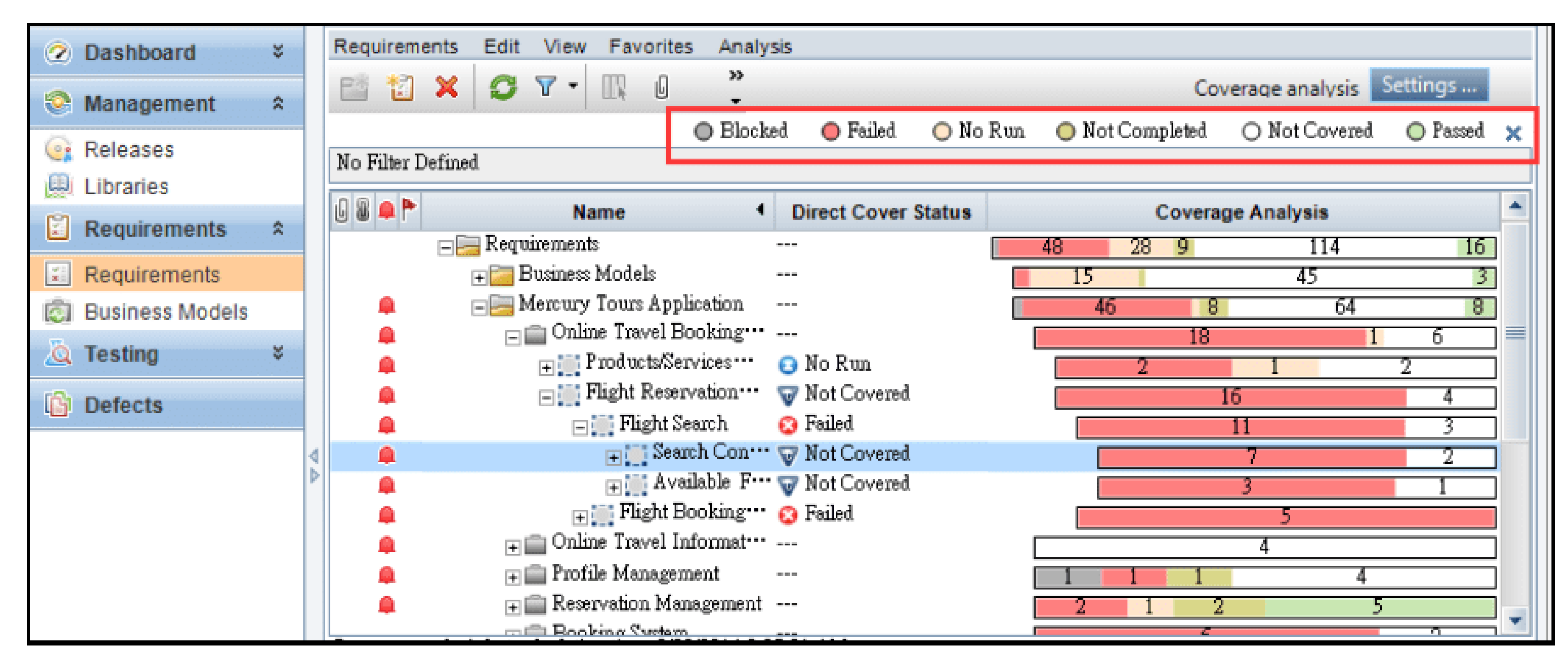


图 18-7　通过周期观察覆盖范围

用户界面元素如表 18-1 所示。

表 18-1　用户界面元素列表

用户界面元素(A-Z)	描　述
覆盖范围分析视图的通用用户界面元素	Coverage Analysis 字段。域的定义，查看“需求模块域” Coverage Analysis 视图菜单和按钮。指令和按钮描述，查看“需求模块菜单和按钮” Coverage Analys 图标。图标描述，查看“需求模块图标”
Lengend	显示用来表示需求和子需求的直接覆盖状态的彩色条码 需求状态列表如下： **Blocked**。被需求覆盖一个或多个测试，具有阻塞执行状态 **Failed**。被需求覆盖一个或多个测试，具有失败状态，且没有一个具有阻塞状态 **No Run**。所有被需求覆盖的测试都具有“未运行”状态 **Not Completed**。被需求覆盖的一个或多个测试具有“未完成”状态的测试，且没有一个具有失败或阻塞状态。另一方面，被需求覆盖的测试都具有“通过”和“未运行”执行状态 **No Covered**。没有与测试连接的需求状态 **Passed**。需求覆盖的所有测试具有通过状态

习题与思考题

1. 如何观察覆盖分析中的说明？
2. 如何设置执行分析的周期？
3. 需求有些什么状态，分别代表什么含义？

练习：覆盖与执行分析

在本练习中，你需要执行下面的任务：

第 1 部分：实现覆盖分析

第 2 部分：实现执行分析

(该练习的指导步骤请参见本章正文)

第19章　使用版本控制

19.1　版本控制概述

在使用版本控制的项目中，可以创建并管理 HP 应用程序生命周期(ALM)实体，包括需求、测试、测试资源、业务流程模型和业务组件。在使用版本控制的项目中对实体做变更时，首先要 check out 实体。当 check out 实体的时候，ALM 会锁定实体，阻止其他用户重写所做的变更。check out 的实体版本对其他用户是可见的。

完成变更之后要 check in 实体，然后，实体的新版本对其他用户可见。可以查看一个实体先前的所有版本或者检验早期的版本。也可以比较两个实体的版本以观察不同版本之间的变化。

注意：为了维护可用性和数据完整性，ALM 存储之前版本的实体，不包括实体间的关系数据。下面这些数据不会保存在前一个实体版本中：需求和测试覆盖、需求可追溯性和缺陷链接。另外，风险数据也不保存在前一个实体版本中。可以在树和网格视图中创建和管理实体。在树视图中，当前用户正在 check out 的实体会显示绿色的锁图标。网格视图包含附加的版本控制域，比如版本状态，表示实体是 check in 还是 Check out 状态。

ALM 有完整的版本控制。你不再需要购买并维护第三方的版本控制系统。相反，ALM 站点管理员可以对每一个项目进行版本控制。版本控制使你可以创建并管理 ALM 的实体，包括需求、测试、组件和测试资源，也可以维护这些实体的前一个版本，如图 19-1 所示。

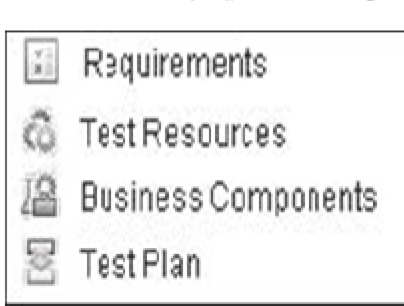
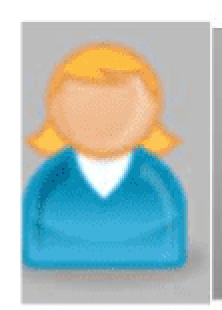


图 19-1　版本控制可创建并管理的 ALM 实体

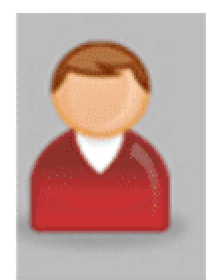
在下列模块中进行版本管理：部门的项目管理员可以在 ALM 站点管理中设置版本控制可用或不可用。

讨论：用户场景

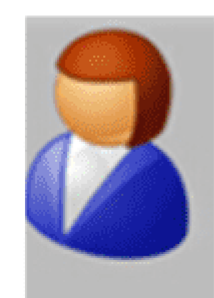
在很多情况下可以用到版本控制，如图 19-2 所示。



玛丽认为需要用两天时间才能完成对下一个需要发布的需求做出必要的变更，她希望在她变更需求期间把所有相关的需求对其他用户签出。



吉姆根据某个需求的变更对5个测试用例也进行了相应的变更。在他完成这个变更之后，该需求被废弃了。吉姆需要再一次把测试用例的前一个版本作为现在的版本。



苏珊发现有一个需求不是按照她的设计进行实施的。当她对这个需求进行审核的时候她发现这个需求被改变了。她想要把这个需求和以前的版本进行比较以找出之前她定义的那个版本。

图 19-2　版本控制使用场景

课堂讨论：这是一次很好的机会，询问班级的同学是否用过版本控制，如果用过，他们能否描述一些场景。讨论使用案例以及版本控制怎样使用。

19.2　如何使用版本控制

1. 前提

检查项目的版本控制是否可用。

2. check out 实体

对版本控制项目中的实体做变更时，实体必须被 check out。用下列方法中的一种：

1) 自动 check out。当编辑一个实体的时候，ALM 显示检查对话框，使你可以 check out 实体。要命令 ALM 在你编辑实体的时候自动 check out 而不显示 check out 对话框，可单击 check out 对话框中的 Do not show again。

2) 手动 check out。选择一个或多个实体，在模块工具栏单击 check out 图标。

3. 撤销 check out

当一个实体在 check out 的时候，可以撤销 check out 来取消变更。右击实体，选择 Versions | Undo Check Out。

4. check in 实体

当变更完成时，check in 实体，创建一个新的、更新过的其他用户可用的版本。

使用下列方法中的一种：

1) check in 单个实体。右键单击实体，选择 Versions | Check In。

2) check in 多个实体。选择实体，在模块工具栏单击 Check In 图标。

5. 查看所有 check out 的实体

可以在当前模块中查看所有已经 check out 的实体，选择实体 check in 或撤销 check out。单击 ALM 窗口右上角的 check in 按钮。

6. 查看历史版本

为了查看和比较，check out 一个实体的先前版本，选择实体，单击 History 标签|Versions 和 Baselines 标签，如图 19-3 所示。

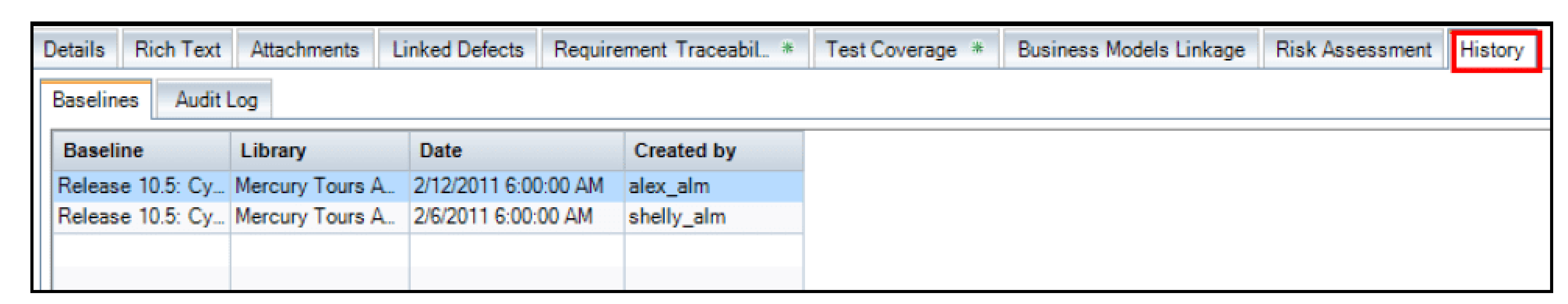


图 19-3　查看和比较一个实体的先前版本

19.2.1　无版本控制的域

使用版本控制的时候，变更会导致 ALM 实体中的一些字段在先前的实体版本中没有保存。在版本控制中，表 19-1 中所列字段的改变不会保存。

表 19-1　Field 列中的字段改变后不会保存

实　体	字　段
Requiements	• Reviewed • Direct Cover Status • Tartget Release • Target Cycle • All RBQM Fields
Tests	Execution Status

19.2.2　使用版本控制

一旦使用版本控制，不管选择需求、测试计划、测试资源或业务组件模块，版本菜单都会出现(如图 19-4 所示)。

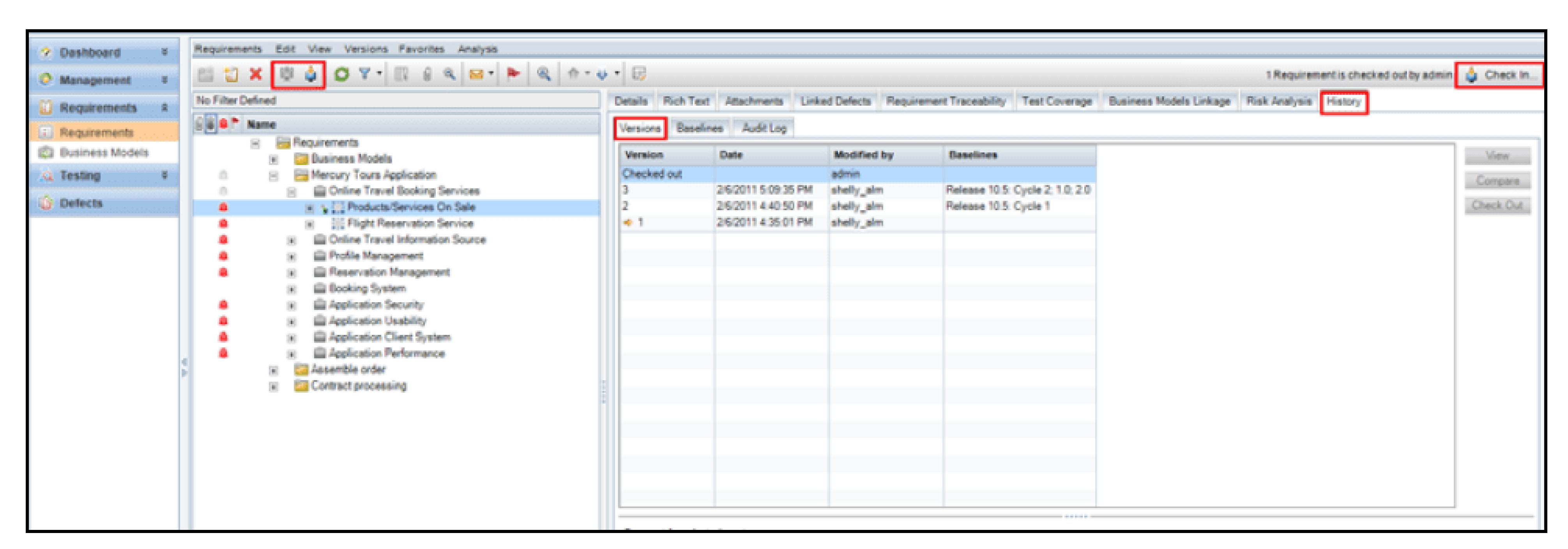


图 19-4　版本菜单视图

19.2.3　版本控制任务

可以在树和网格视图中创建和管理实体。在树视图中，当前用户正在 check out 的体会显示一个绿色的锁图标。网格视图包含附加的版本控制域，比如版本状态，表示实体是 checked in 或 checked out 状态。

注意，下面提到的字段在版本控制中不会保存：

- 字段：评估、直接覆盖状态、目标发布、目标循环和所有的 RBQM 字段。
- 测试字段：执行状态。

19.3　check out 和 check in

在开启版本控制的项目中，只能通过 check out 来做变更。有两种 check out 实体的方式：手动 check out 实体和自动 check out 实体。

19.3.1　手动 check out 实体

手动 check out 实体的步骤如下(参见图 19-5)：

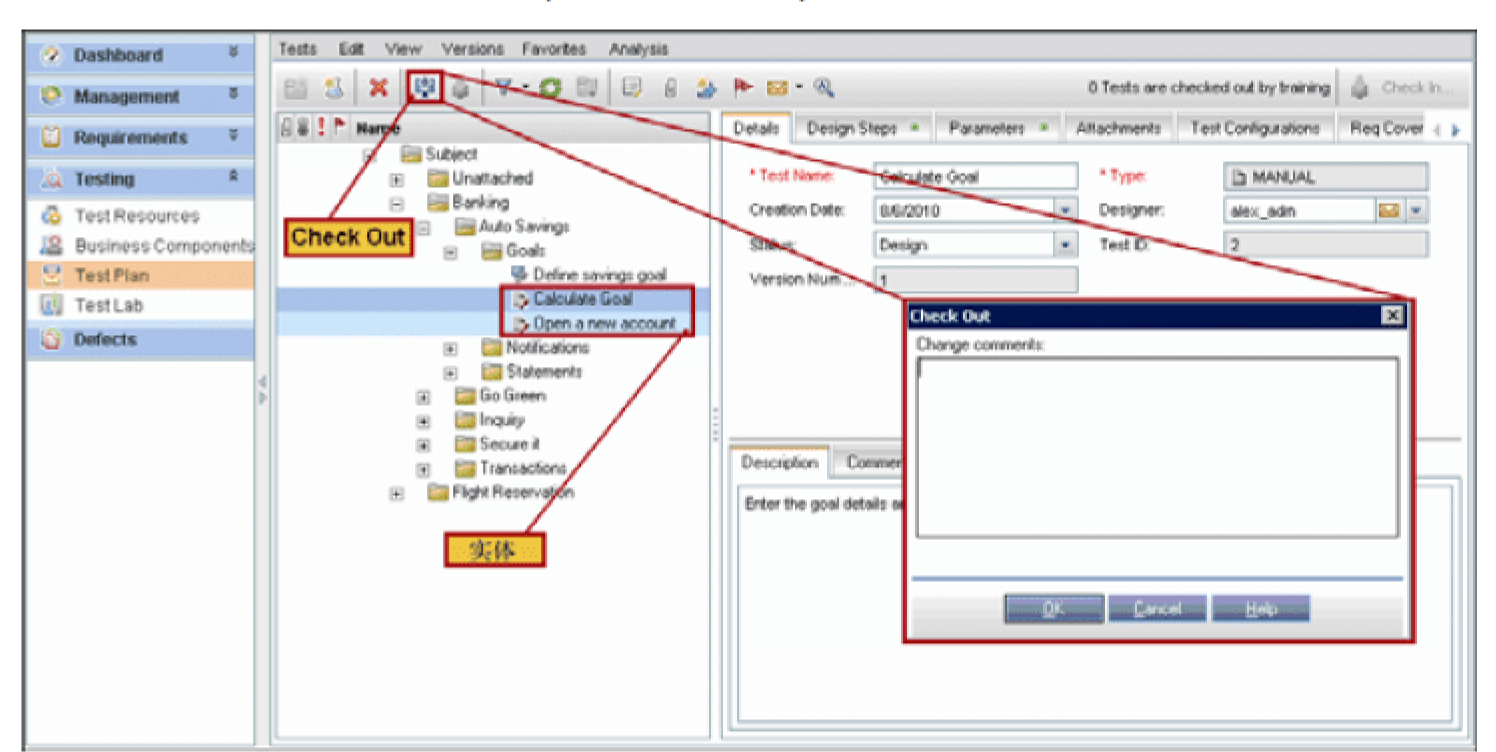


图 19-5　手动 check out 实体的步骤

(1) 选择要 check out 的实体，如果要选择多个实体，按住 Ctrl 键。

(2) 单击 check out 按钮(也可以右击实体，选择 Versions|Check Out)，打开 Check Out 对话框。

(3) 在 Change Comments 栏，输入 Check Out 原因的简要说明。

(4) 单击 OK。在 Test Plan 树视图中，实体的前面会显示一个锁图标，表明正在 check out 这个实体。

19.3.2　自动 check out 实体

如果在版本控制中已经开始对一个实体做变更，将会提示你 check out 实体。

要自动 check out 实体，执行下面的步骤：

(1) 选择实体，开始编辑。如图 19-6 所示，打开 check out 对话框。

(2) 在 Change Comments 框中输入 check out 原因的简要描述。

(3) 单击 OK，在树视图中，实体会显示一个绿色的锁图标，表示正在被 check out。

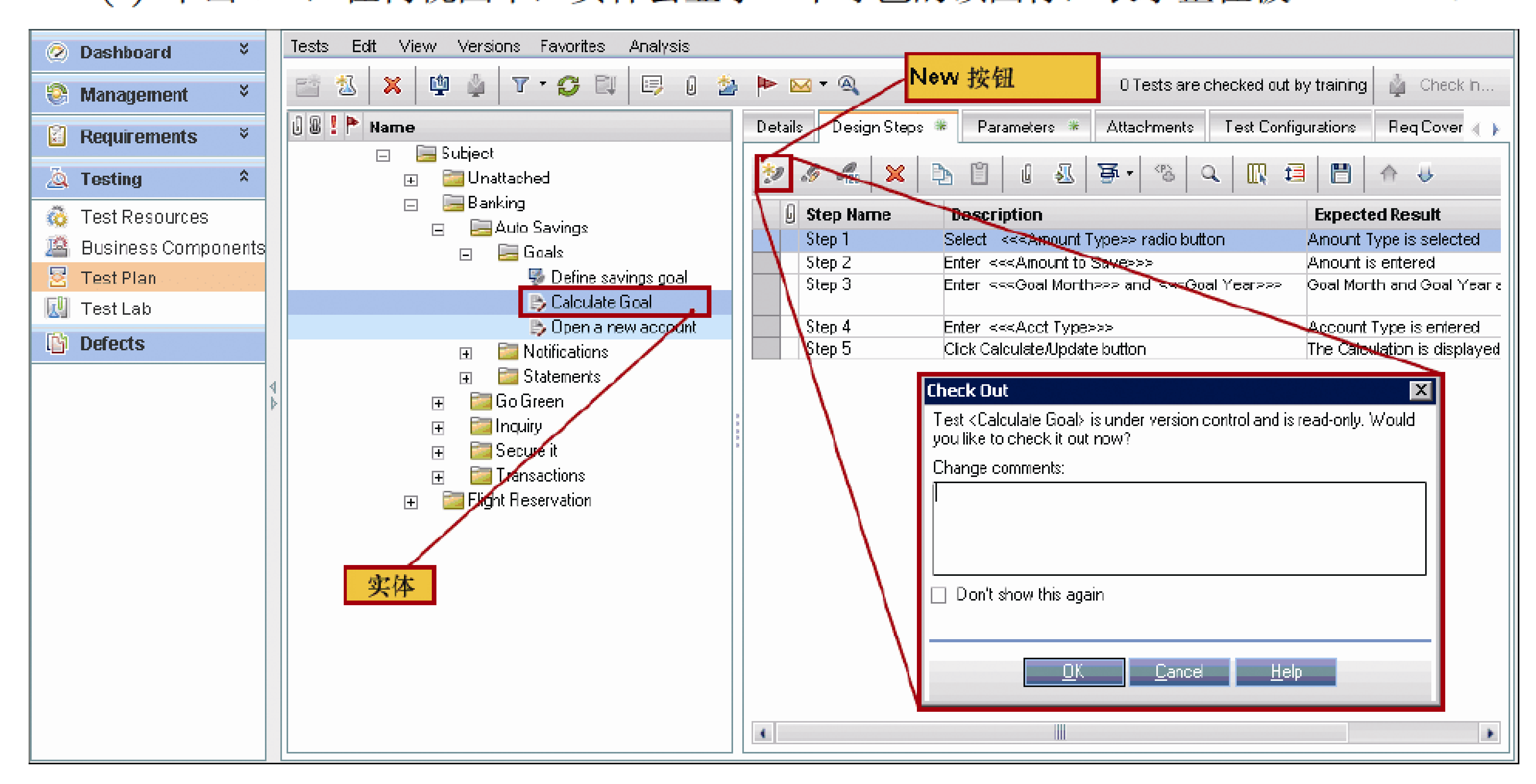


图 19-6　自动 check out 实体的步骤

19.3.3　撤销 check out

在 check out 的时候可以通过撤销 check out 来取消变更。为了使其他用户可以撤销实体的 check out，你必须有适当的用户权限。在撤销 check out 的时候，对无版本控制的域所做的变更不会撤销，新值依然存在。

为了撤销一个 check out，执行下面的步骤(如图 19-7 所示)：

(1) 选择一个或按住 Ctrl 键选择多个要撤销 check out 的实体。

(2) 选择 Versions | Undo Check Out 或右击实体，选择 Versions | Undo Check Out。

(3) 单击 OK 以确认。

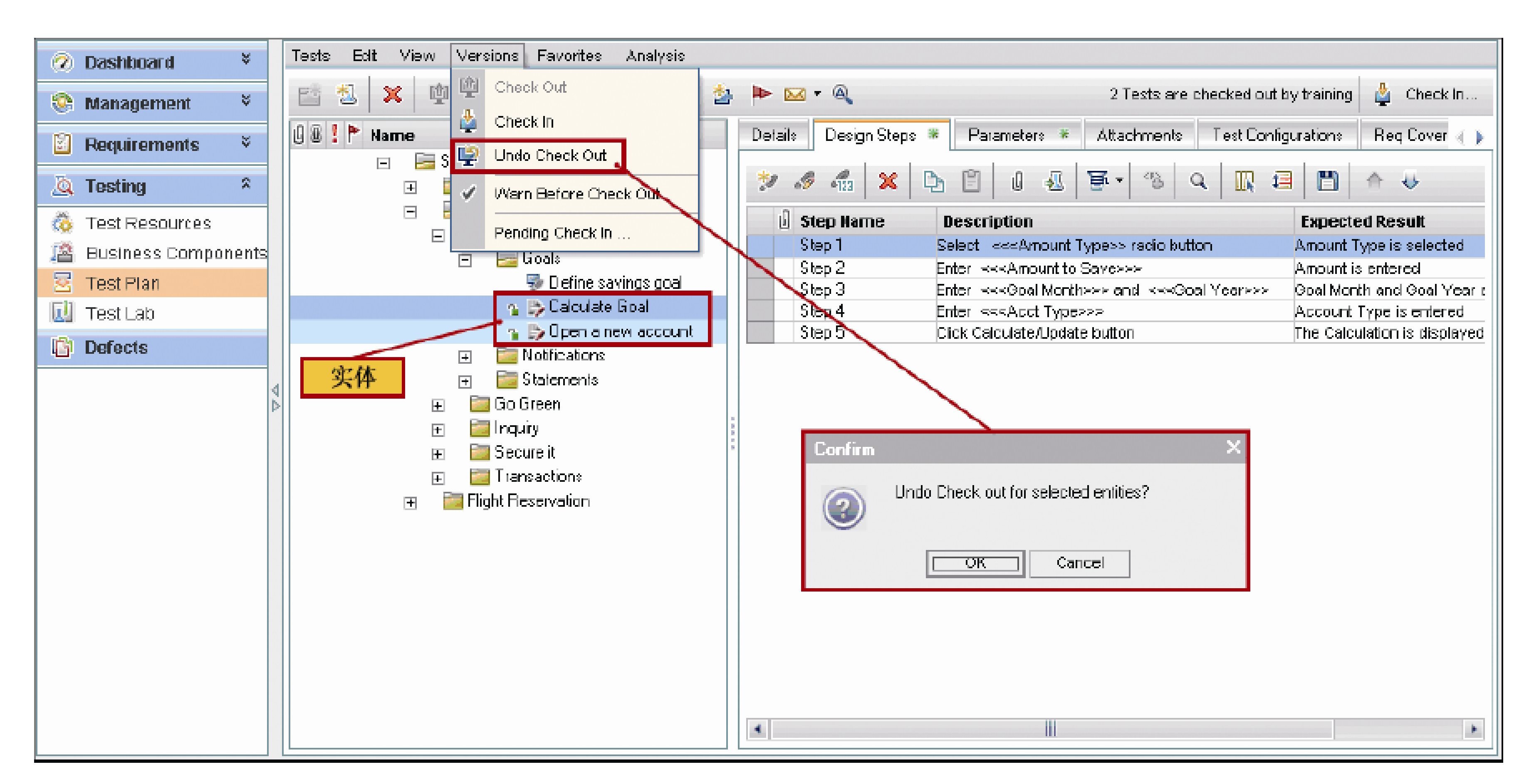


图 19-7　撤销 check out 的步骤

19.3.4　check in 实体

对实体做完变更之后，check in 实体。每次 check in 实体的时候就会创建一个新版本。例如，当前一个需求的版本编号是 2，你检查需求并做了一些变更。在把需求 check in 的时候，ALM 把它的版本定义为 3。实体 check in 的时候，实体是解锁的而且其他用户是可用的。

要 check in 实体，执行下面的步骤(如图 19-8 所示)：

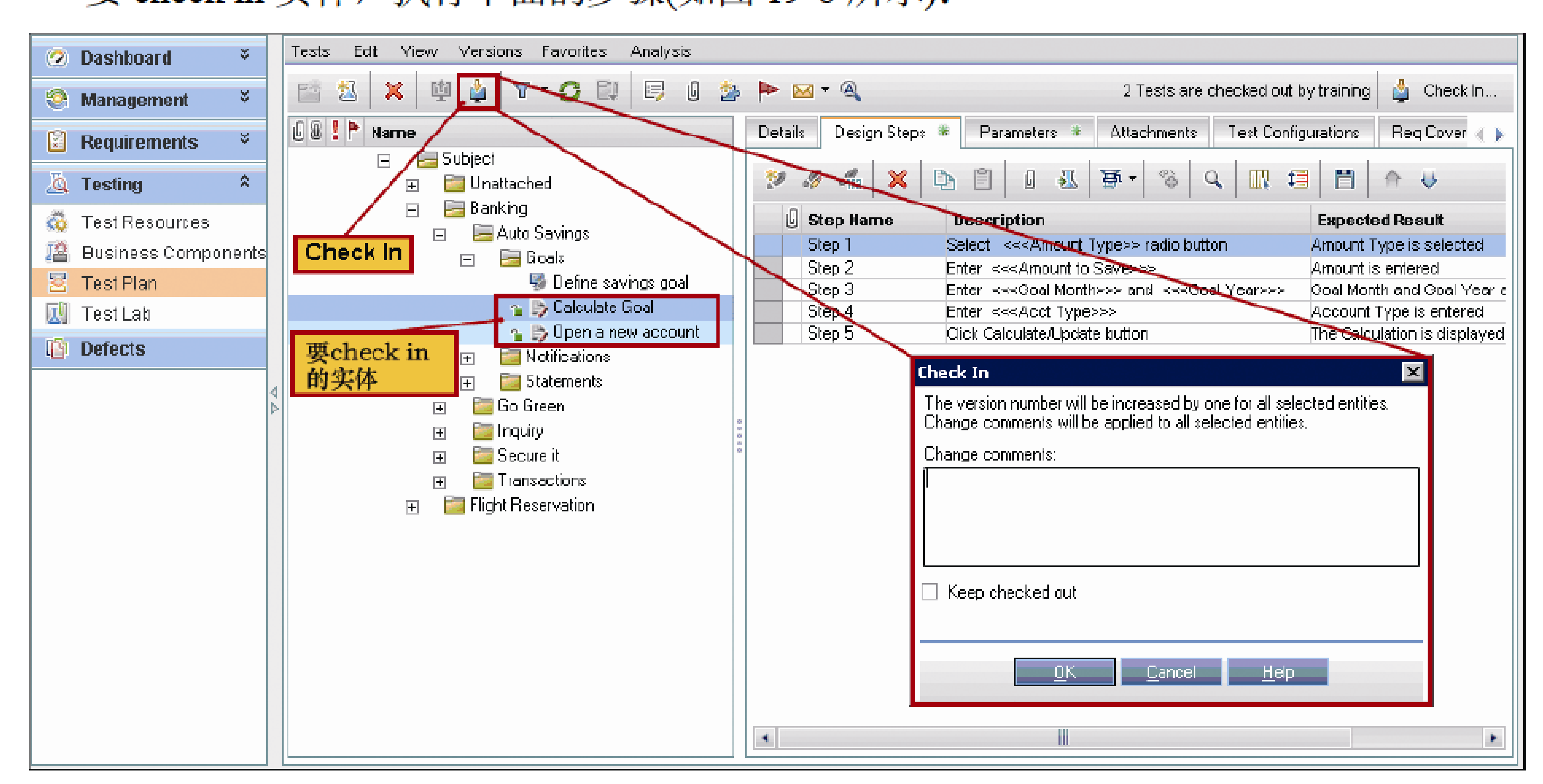


图 19-8　check in 实体的步骤

(1) 选择要 check in 的实体，如果要 check in 多个实体，按住 Ctrl 键并选择多个要 check in 的实体。

(2) 单击 Check In 按钮，也可以右击实体，选择 Versions | Check In，打开 Check In 对话框。

(3) 在 Change Comments 框中输入这次版本变化的简要描述。

在保持实体 check out 的时候可以选择 Keep Checked Out，从而在新版本中保存变更。

19.3.5　查看 check out 实体

可以查看当前模块的所有已经 check out 的实体列表，选择实体进行 check in 或撤销 check out。要查看 check out 的实体，执行下面的步骤(如图 19-9 所示)：

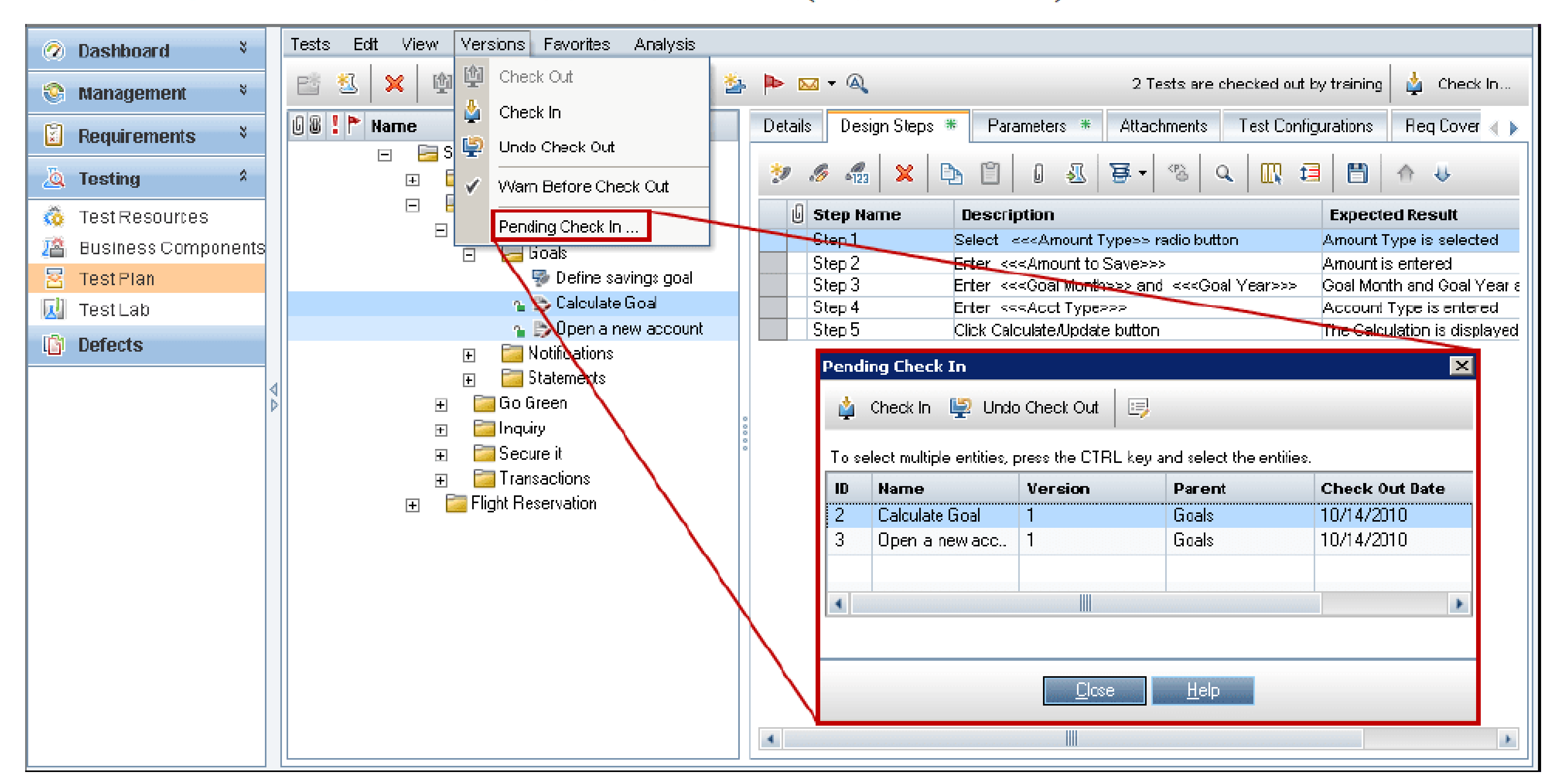


图 19-9　查看 check out 实体的步骤

(1) 选择 Versions | Pending Check In(或者单击右边的模块工具中的 Check In 按钮)，打开 Pending Check In 对话框，显示当前模块中被当前用户 check out 的所有实体的列表。

(2) 要 check in 一个实体，在实体列表中选择一个实体或按住 Ctrl 键选择多个实体，单击 Check In 按钮。

(3) 如果要撤销 check out，从列表中选择一个实体或按住 Ctrl 键选择多个，单击 Undo Check Out。

19.4　查看版本历史

可以查看一个实体的历史记录，包括先前的所有版本，创建每个版本的用户名和创建日期。也可以比较两个版本，或者 check out 之前的版本。还可以查看版本存储中的基线(基线将会在下一章讲到)。

要查看版本的历史，执行下面的步骤(如图 19-10 所示)：

(1) 选择一个实体的单击 History 标签，会显示该实体的版本历史。在 Versions And Baselines 标签中，在 View By 框中选择 Versions。实体的版本历史会在表 19-2 中显示。

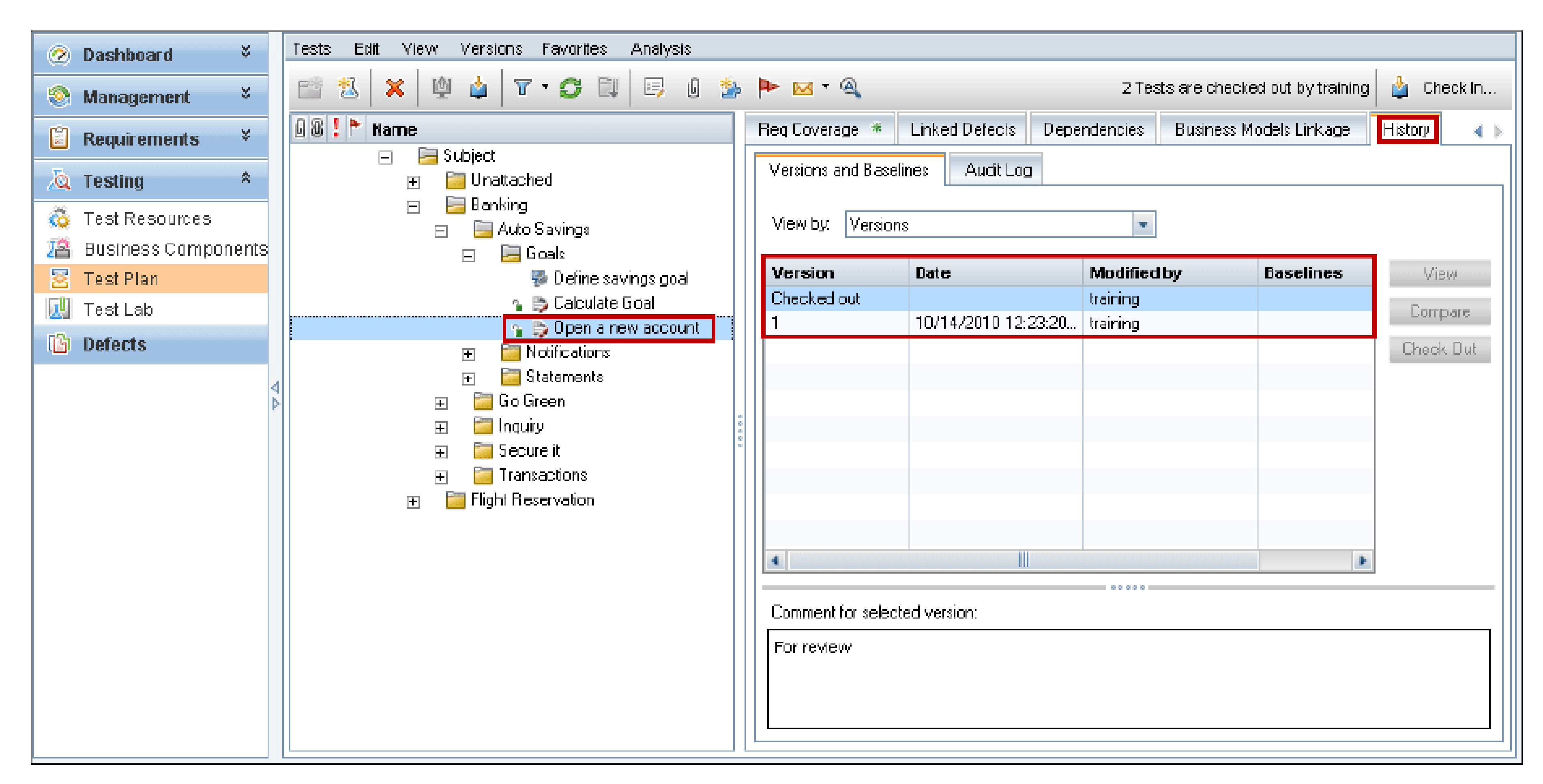


图 19-10　查看一个实体的历史记录的步骤

表 19-2　版本历史信息

列	描　述
Version	版本号。如果实体当前已签出，那么用于显示签出版本的 Version 列显示为“签出”
Date	版本创建的日期
Modified By	创建版本的用户
Baseline	版本出现时所在的基线

(2) 在 Comment For Selected Version 下，查看用户在 Check In 版本的时候写的说明。

(3) 为了查看之前版本的详细说明，选择版本并单击 View，打开 Test Details 对话框，显示只读的版本详细说明。例如，可以查看一个测试的先前版本的详细说明，如图 19-11 所示。

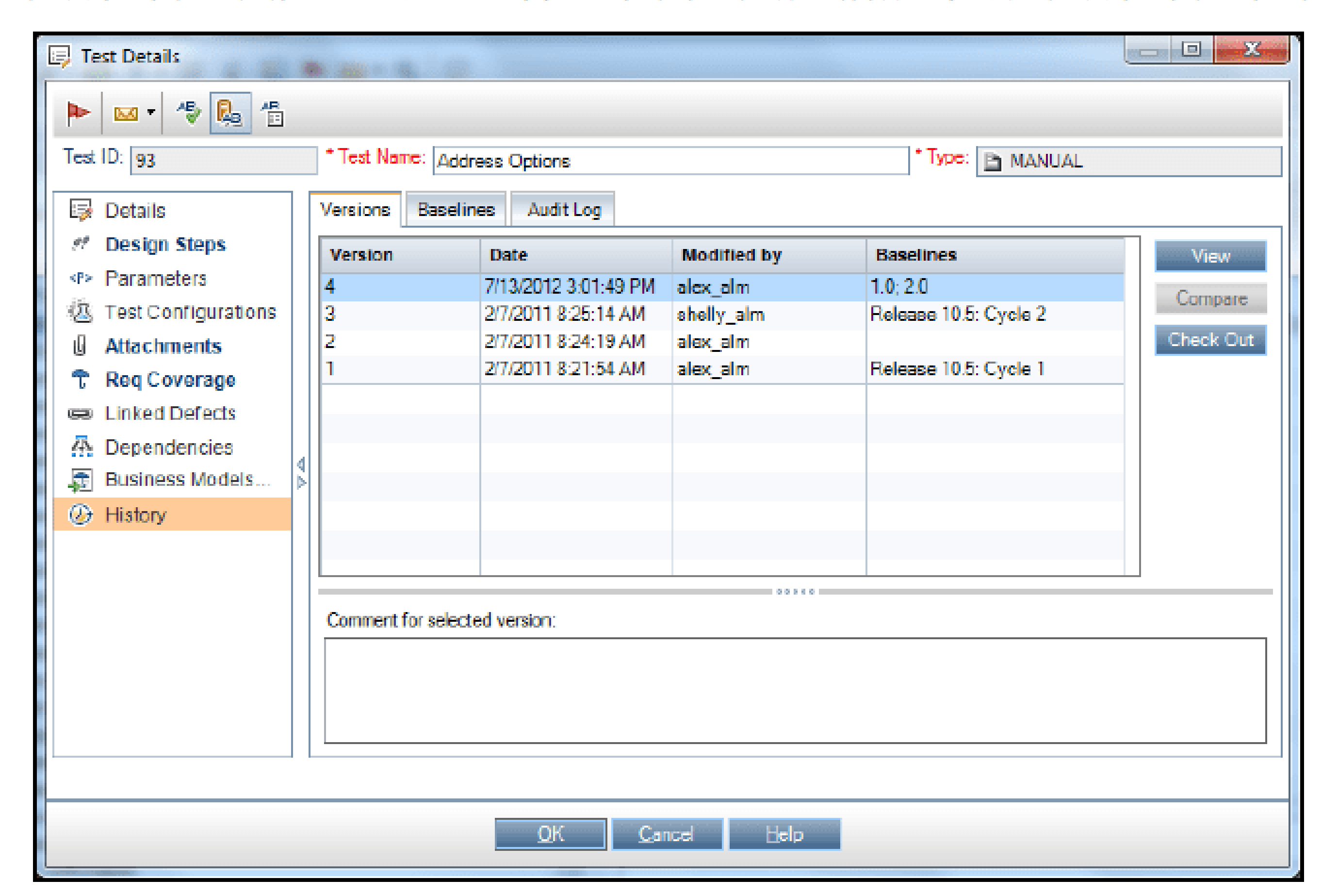


图 19-11　版本创建时的详细说明

(4) 在侧边工具栏中单击按钮以查看版本的附加说明，比如 Design Steps、 Parameters 和 Attachments。按钮是否有效，取决于版本控制中存储实体类型的数据。单击 OK 以关闭 Test Details 对话框。

19.5　版本比较

为了比较两个版本，按住 Ctrl 键并选择每个版本，单击 Compare，如图 19-12 所示。

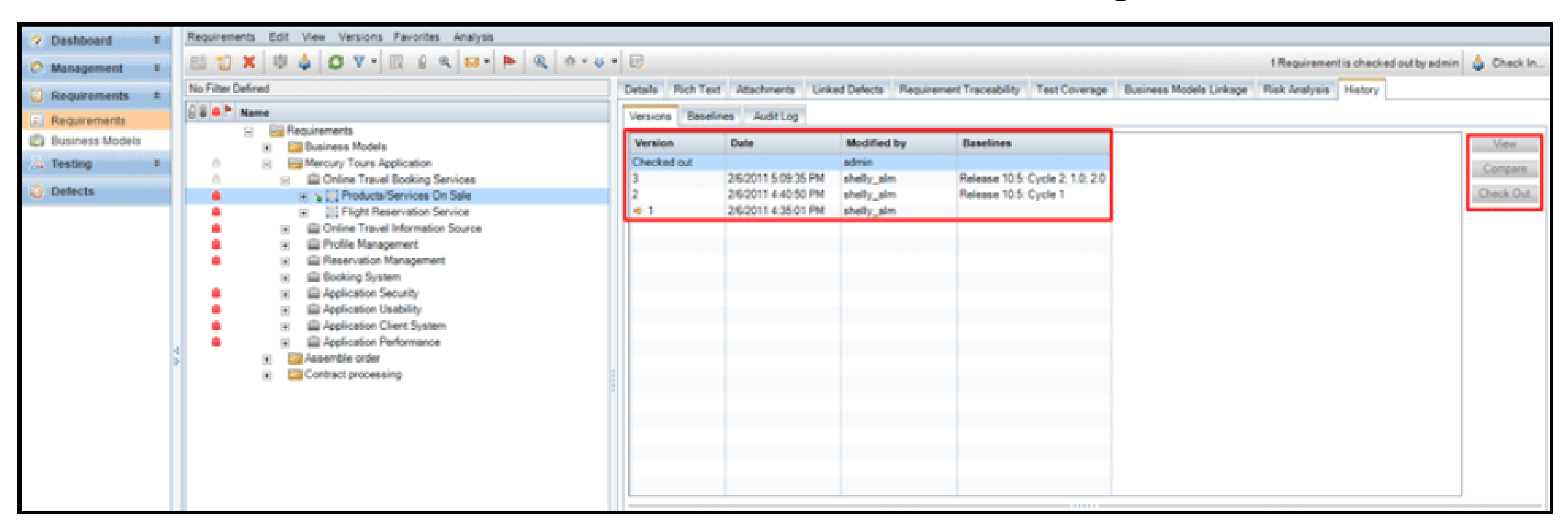


图 19-12　版本比较界面

19.5.1　比较两个版本的例子

产品经理发现产品开发及执行过程和他期望的不同，他检查产品需求，发现有些地方做了更改，他比较当前的需求版本和发布之前已经确定的需求。

19.5.2　恢复早期的版本

一名 QA 测试人员收到一个新的刚开发好的银行应用程序版本。他开始更新相关的测试以满足新发布版本的需求。开发团队发出通知，这个版本中有一个重大的问题，开发团队回退到之前的版本。测试人员决定 check out 并恢复到之前应用程序的测试版本并继续从之前那个测试版本测试。

19.5.3　锁定实体

一名业务分析人员想要更新一个应用程序的功能，这样他需要更新一系列需求，这需要几天的时间来更新需求，而且在他编辑需求的时候不希望其他人做任何改变。他 check out 相关的需求，开始编辑。

19.6　升级旧版本

为了 check out 之前的版本，单 Check Out，打开确认对话框，单击 OK 确认，如图 19-13 所示。

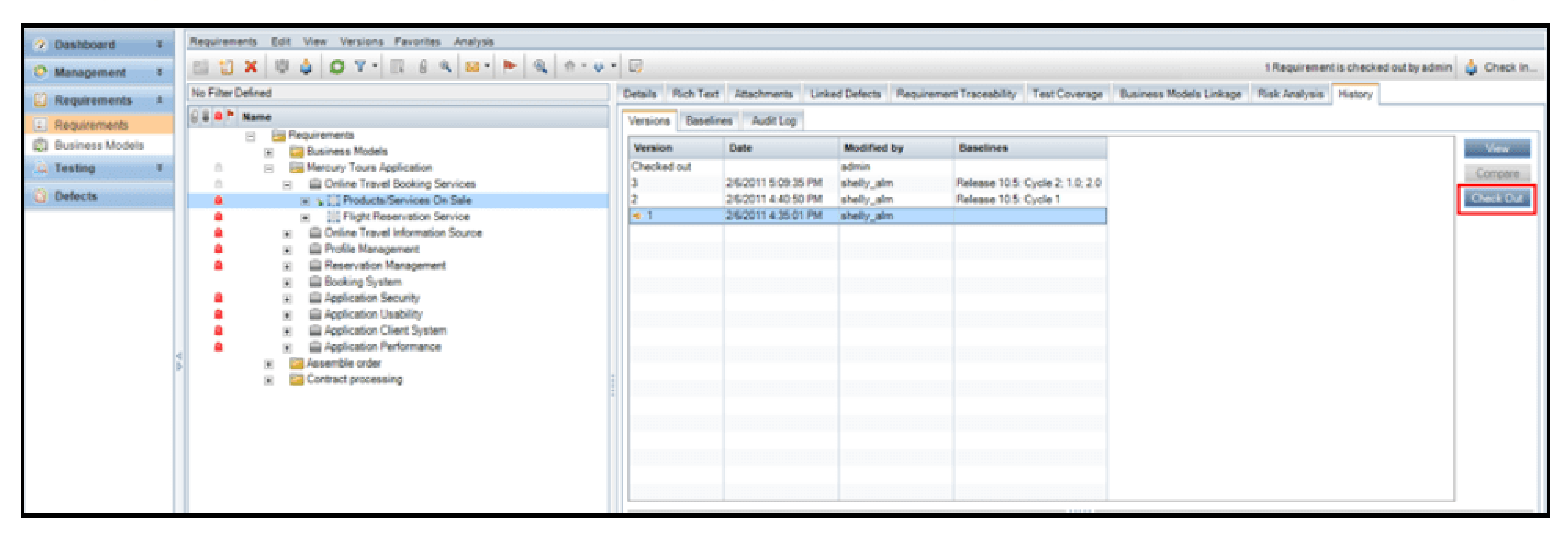


图 19-13　check out 旧版本

注意，为了维护可用性和数据完整性，ALM 保存实体先前的版本时，不会保存实体之间的关系的大多数数据。先前的实体中不会保存下面这些数据：需求和测试覆盖、需求可追溯性和缺陷链接。另外，风险数据也不会保存。

习题与思考题

1. 什么样的实体可以使用版本控制？
2. 在版本控制的项目中，要对实体做变更，首先要做什么？

1) 将 Version Control 设置成 off。

2) check in 实体。

3) check out 实体。

4) 更新实体。

3. 如果想把一个实体恢复到之前的版本，你需要采用下面哪一种方式？

1) check in 实体。

2) check out 实体。

3) 更新实体。

4) 改变实体的历史记录。

练习：使用版本控制

在这个练习中，你将打开一个现有项目，这个项目的版本控制已经被管理员允许使用。你要完成下面的任务：

第 1 部分：check out 实体

第 2 部分：check In 实体

第 3 部分：版本比较

第 4 部分：升级旧版本

(该练习的指导步骤请参见本章正文)

第20章　Library管理

20.1　Library 概述

在 Library 模块中，通过定义层次化结构的 Library 树创建和管理 Library。一个 Library 表示项目的一系列实体和它们之间的关系。Library 中的实体可以包含需求、测试、测试资源和业务组件。创建 Library 之后，可以通过创建基线来随时记录项目的变化。基线是在特定时间点时 Library 的一个快照。可以对比在应用开发生命周期内各个阶段的基线。查看基线的历史记录可以让你随时跟踪 Library 中单个实体的变化。当开发继续时，可以查看并比较存储在基线里的实体的所有版本。也可以导入一个 Library，导入 Library 可以重新使用或共享已存在的一系列实体。当继续开发时，可以对比并同步 Library。本节描述的 Library 模块功能在 ALM 初始版本和企业版本中导入 Library 功能时是不可用的。

20.1.1　Library 模块

可以对比应用程序生命周期管理过程的各个阶段的基线。例如，可以通过对比一个 Library 中的两个基线以随时查看 Library 中测试的变化。在 ALM 中运用 Library 模块定义 Library。要显示这个模块，在侧边栏上单击 Management 按钮，然后单击 Libraries 选项卡，如图 20-1 所示。

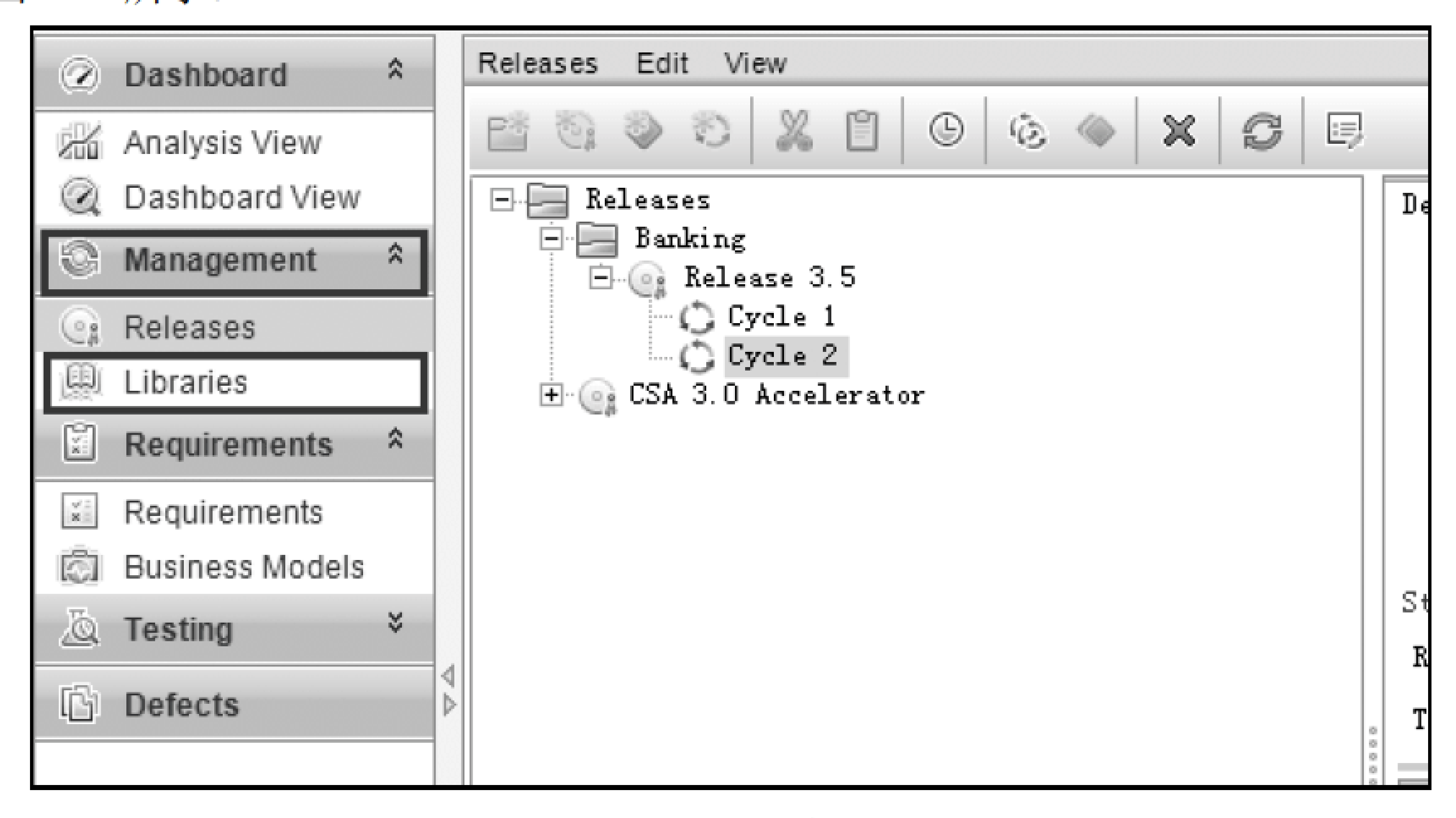


图 20-1　库模块界面

20.1.2　Library 菜单栏

Library 菜单栏包含以下子菜单(如图 20-2 所示)：

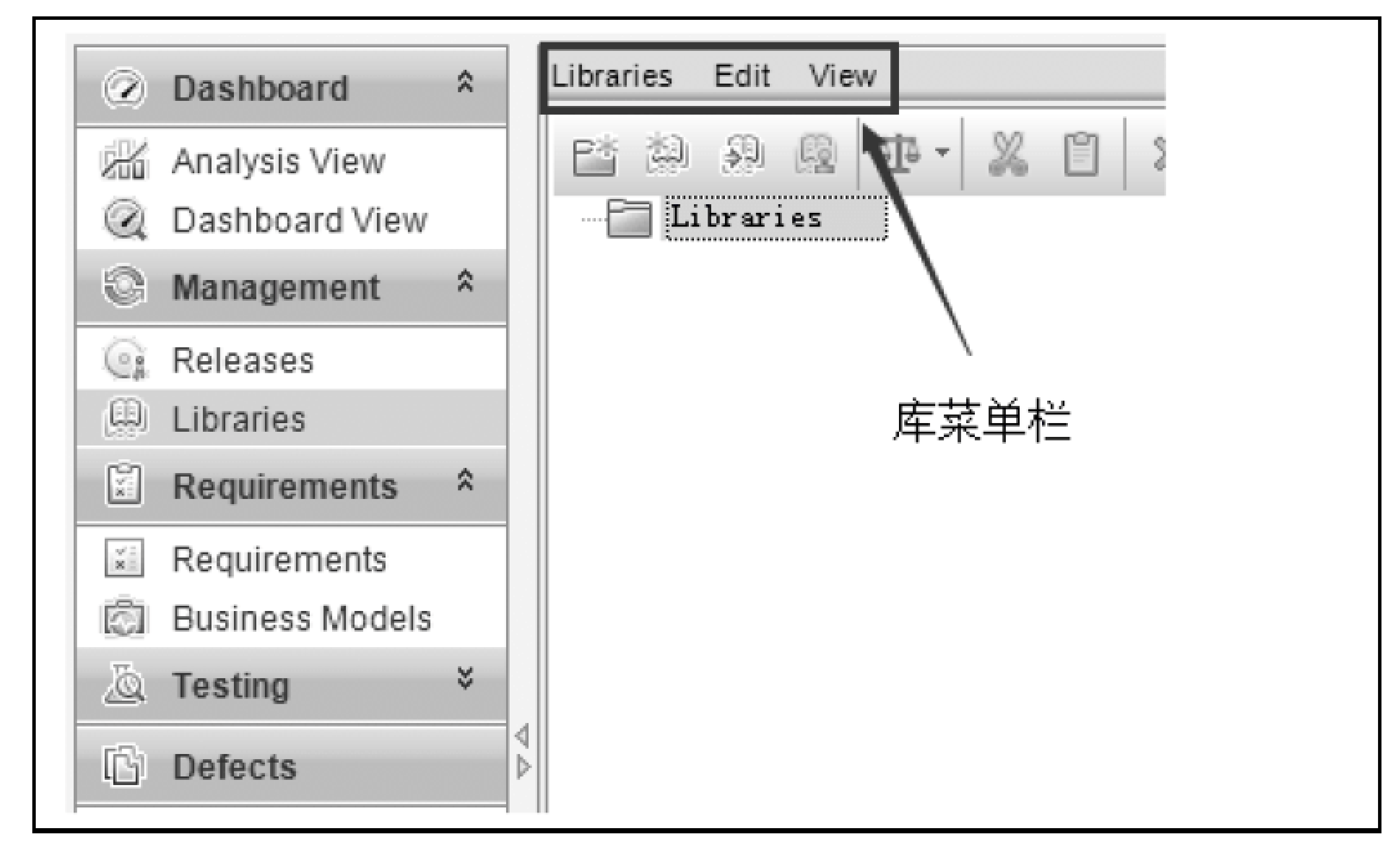


图 20-2　Library 菜单栏

1) Library 子菜单包含添加 Library 文件夹、库和基线的命令，也包含对比库和基线的命令。

2) Edit 菜单包含 Cut、Paste、Delete、Rename Library Folders、Library 和 Baselines 命令。

3) View 菜单可以用于刷新 Library 树，也可以过滤 Library 树中的项目或清除已存在的过滤器。

20.1.3　Library 工具栏

Library 工具栏包含如下按钮(如图 20-3 所示)：

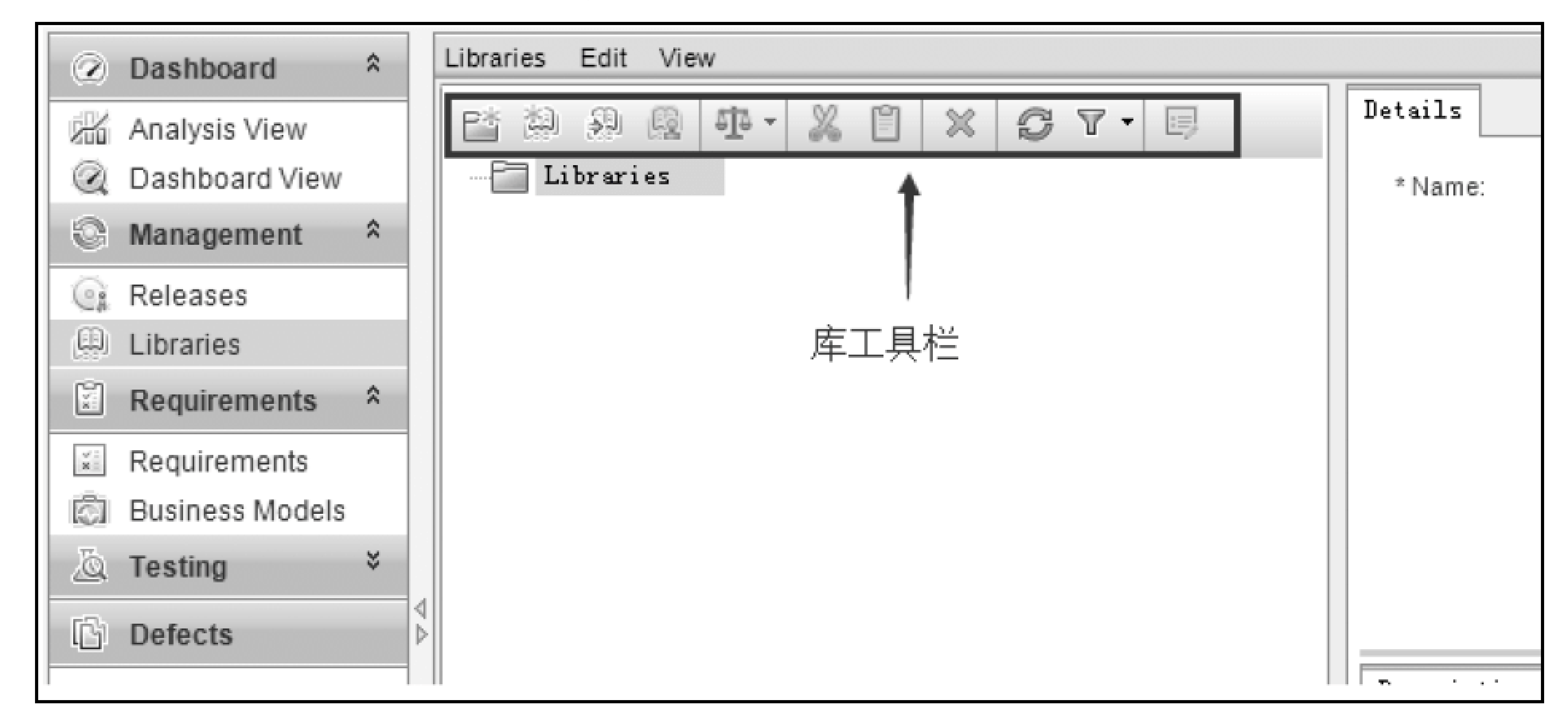


图 20-3　Library 工具栏

1) New Folder：在 Libraries 树中添加新的文件夹。

2) Create Library：在 Libraries 树中添加新的库。

3) Import Library: 导入新的库并将它添加到 Library 树中。

4) Create Baseline: 创建一个基线并添加到 Libraries 树中。

5) Compare To: 将当前选择的库或基线与其他库或基线进行对比。

6) Cut:从 Libraries 树中剪切 Library 文件夹或库并将其移到 Libraries 树的其他位置。

7) Paste: 在 Libraries 树中指定的位置粘贴已剪切的 Library 文件夹或库。

8) Delete: 删除选择的条目。删除 Library 文件夹将删除它里面的库和基线。删除库也将删除它的基线。

9) Refresh: 刷新 Libraries 树和标签,显示最新的数据。

10) Filter:过滤 Libraries 树中的库。

11) Library Details:打开 Library Details 对话框，查看并编辑所选库的详细信息。

20.2　创建 Libraries 树

通过创建 Libraries 树为库定义层次结构框架，Libraries 树可以包含文件夹和子文件夹。创建 Libraries 树的步骤如下(如图 20-4 所示):

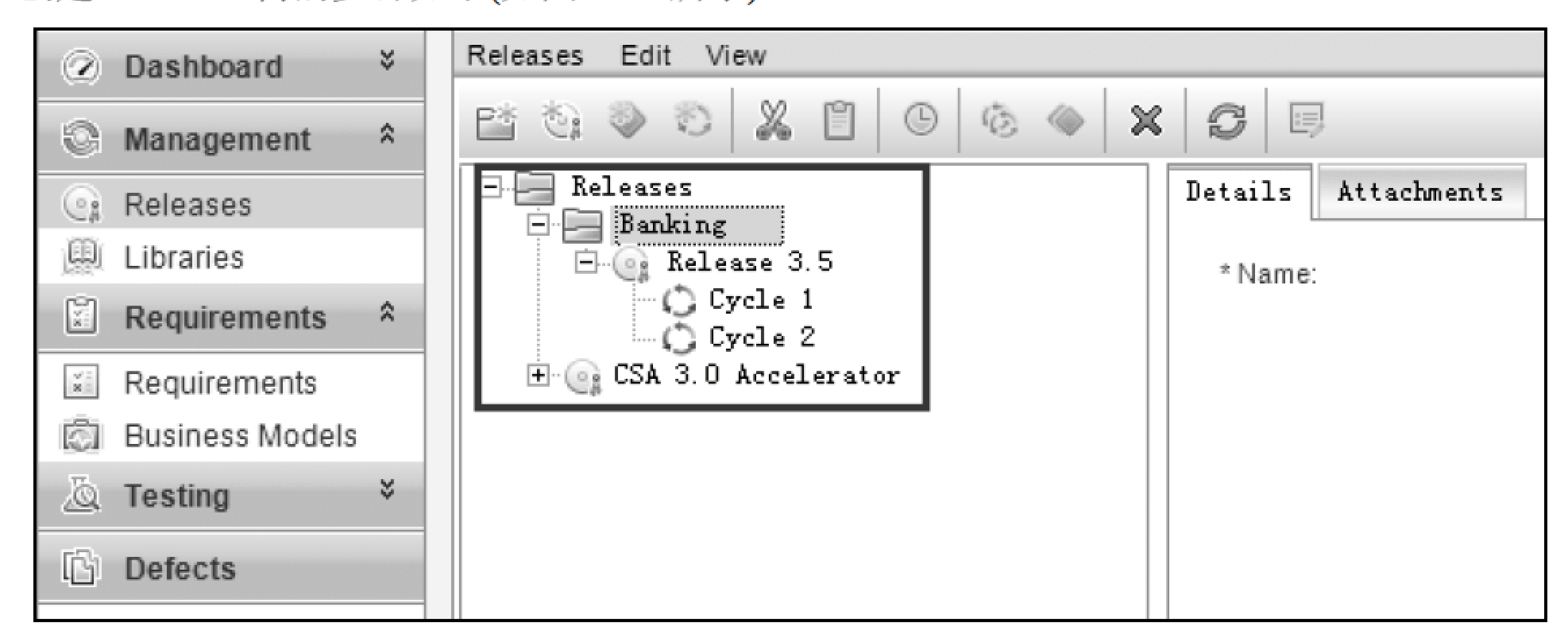


图 20-4　Libraries 树

(1) 在 Libraries 模块中，选择 Libraries 文件夹。

(2) 单击 New Folder 按钮，或者选择 Libraries | New Folder，弹出 New Libraries Folder 对话框。

(3) 在 New Libraries Folder 对话框中，输入文件夹的名称并单击 OK，新的文件夹被添加到 Libraries 树中。

(4) 在 Details 对话框的 Description 面板中，输入 Library 文件夹的描述信息，如图 20-5 所示。

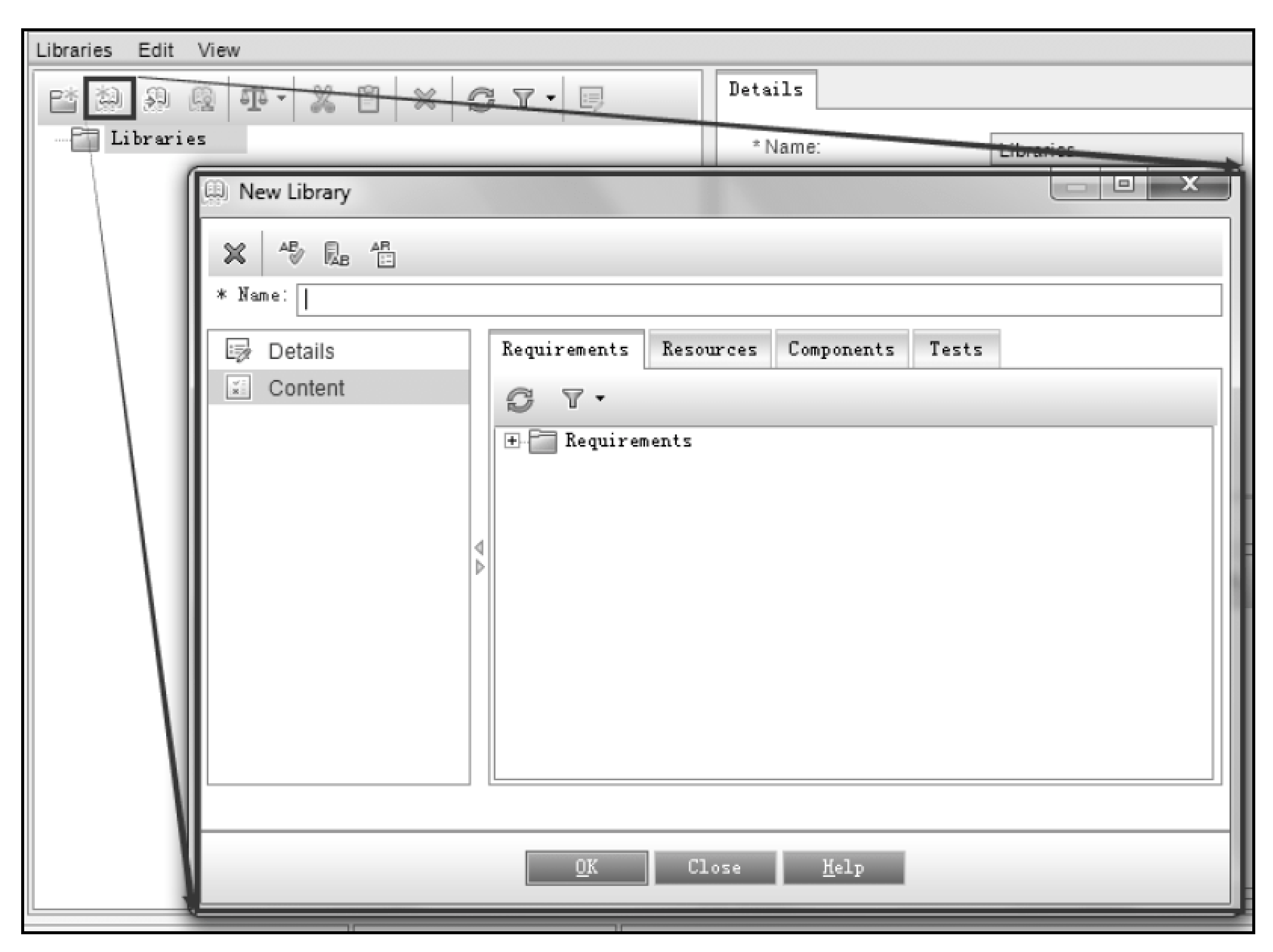


图 20-5　输入 Libraries 树的描述信息

20.2.1　创建库

创建 Libraries 树以后，向 Libraries 文件夹中添加库，如图 20-6 所示。可以创建一个新的库，或者导入存在的库。向 Libraries 文件夹中添加库的步骤如下：

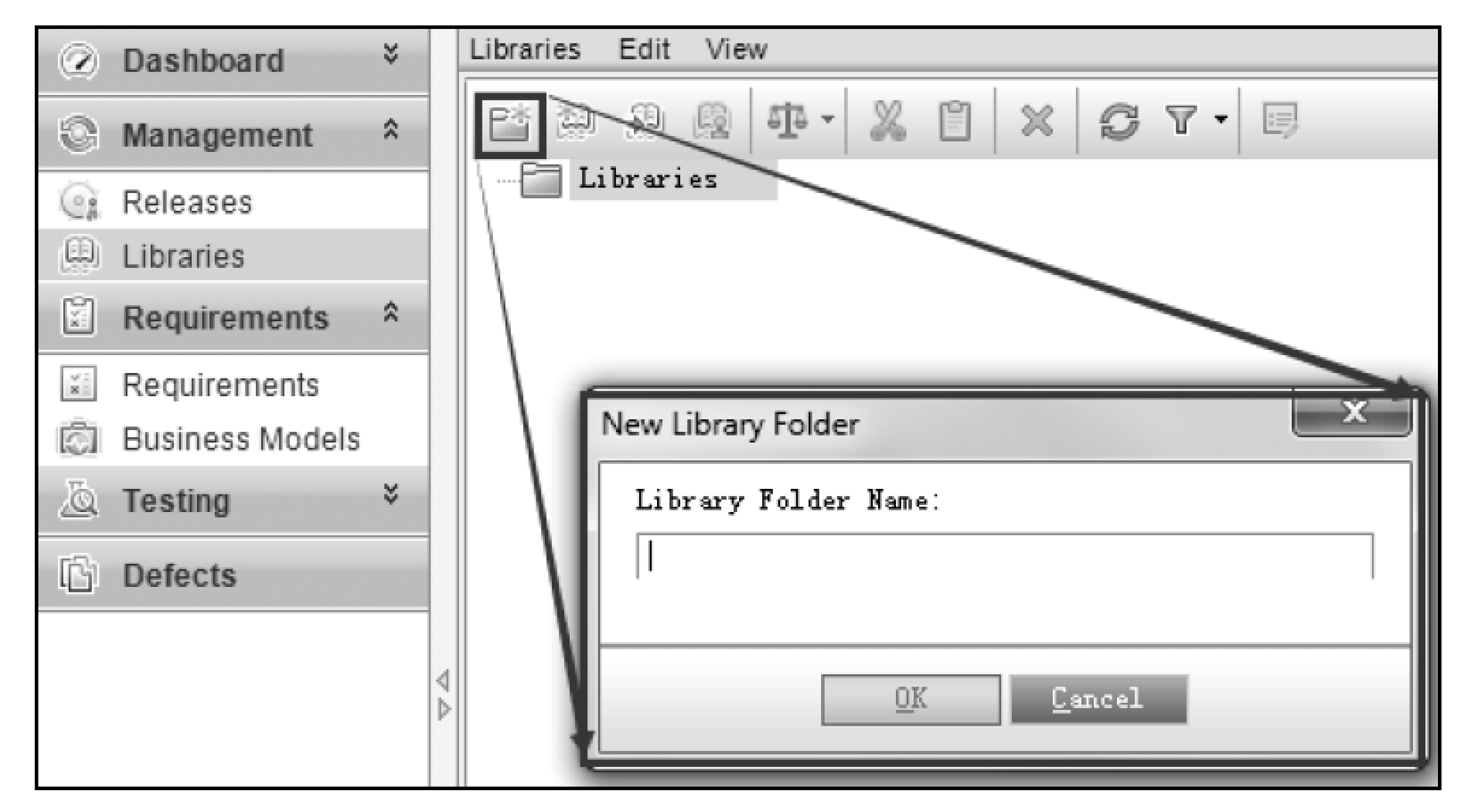


图 20-6　创建库

(1) 在 Libraries 子菜单中，选择 Libraries 文件夹。

(2) 单击 Create Library 按钮，或者选择 Libraries | Create Library。

(3) 弹出 New Library 对话框。

(4) 输入 Name。

(5) 单击 Content 页面。

(6) 打开 Requirements 选项卡，展开 Requirements 树目录，选择正确的文件夹并单击 OK。

(7) 在 Resources、Components 和 Tests 选项卡中选择正确的文件夹。

(8) 在 Tests 选项卡中选择 Tests Covering Selected Requirements，然后选择能覆盖在 Requirements 选项卡中所选需求的测试。

20.2.2　实体过滤

创建库之后，可以为库中的每个实体树定义一个过滤器，这可以更好地控制 Libraries 目录中的内容，如图 20-7 所示。

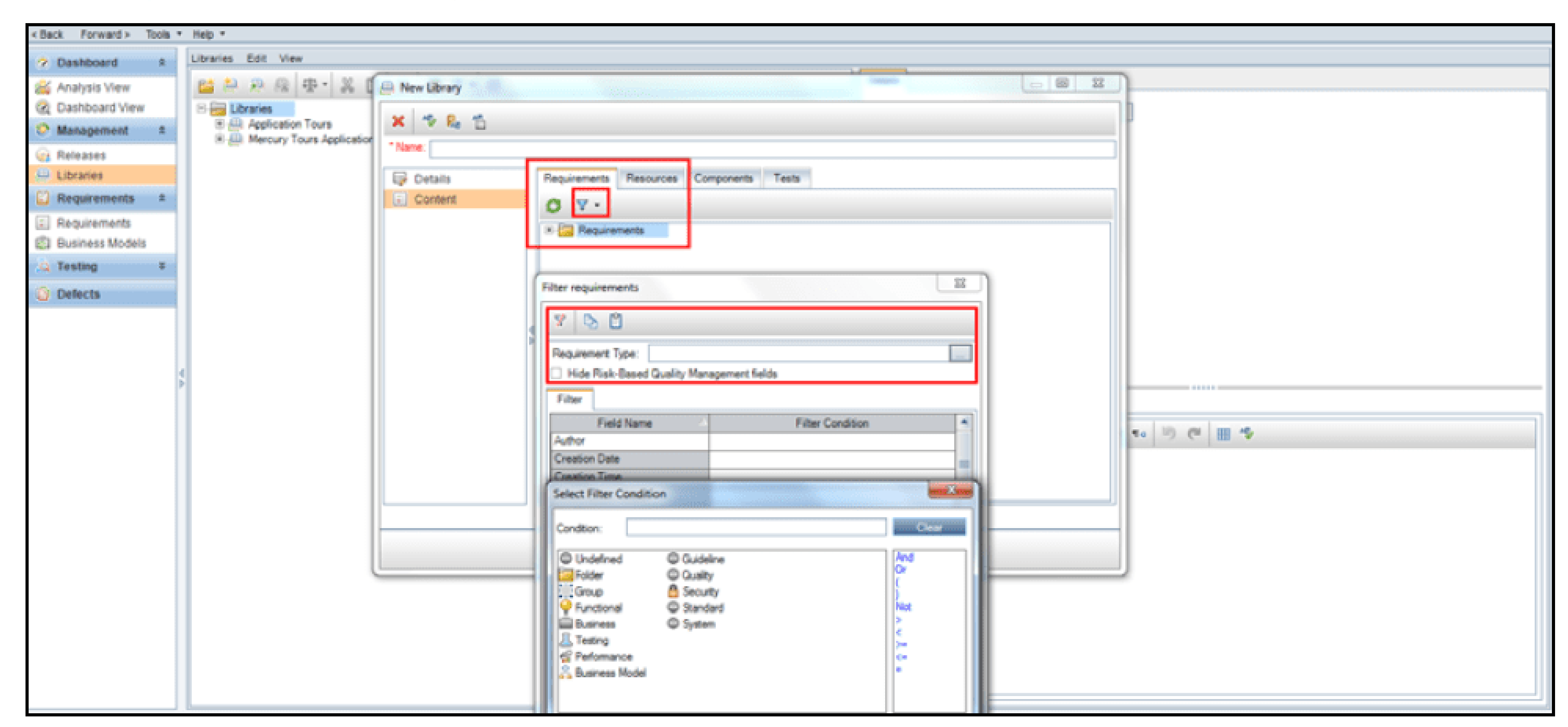


图 20-7　进行实体过滤

20.2.3　Content 选项卡

Content 选项卡可用于选择或查看库中包含的实体，如图 20-8 所示。

为实体(需求、资源、组件或测试)树定义过滤器的步骤如下：

(1) 单击 Filter/Sort 按钮。

(2) 弹出 Filter Entities 对话框。

(3) 单击 Filter Condition 下拉菜单，弹出 Select Filter Condition 对话框。

(4) 设置条件值并单击 OK。

在 Filter Entities 对话框中单击 OK 按钮，满足过滤条件的实体显示在实体树中。

添加覆盖需求测试，如图 20-9 所示。

当在库中选择需要获取的测试时，有自动在库中获取那些覆盖需求的测试的选项。在库中获得覆盖需求的测试的步骤如下：

(1) 单击 Content 选项卡。

(2) 单击 Tests 选项卡。

(3) 选择 Tests Covering Selected Requirements 选项。

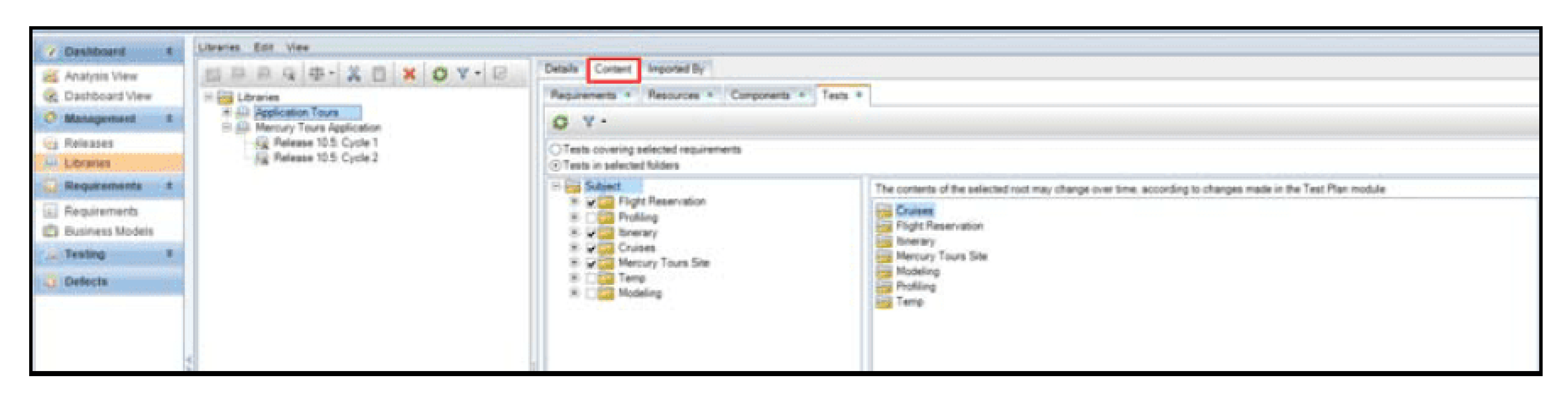


图 20-8　在 Content 选项卡中选择的库所包含的实体

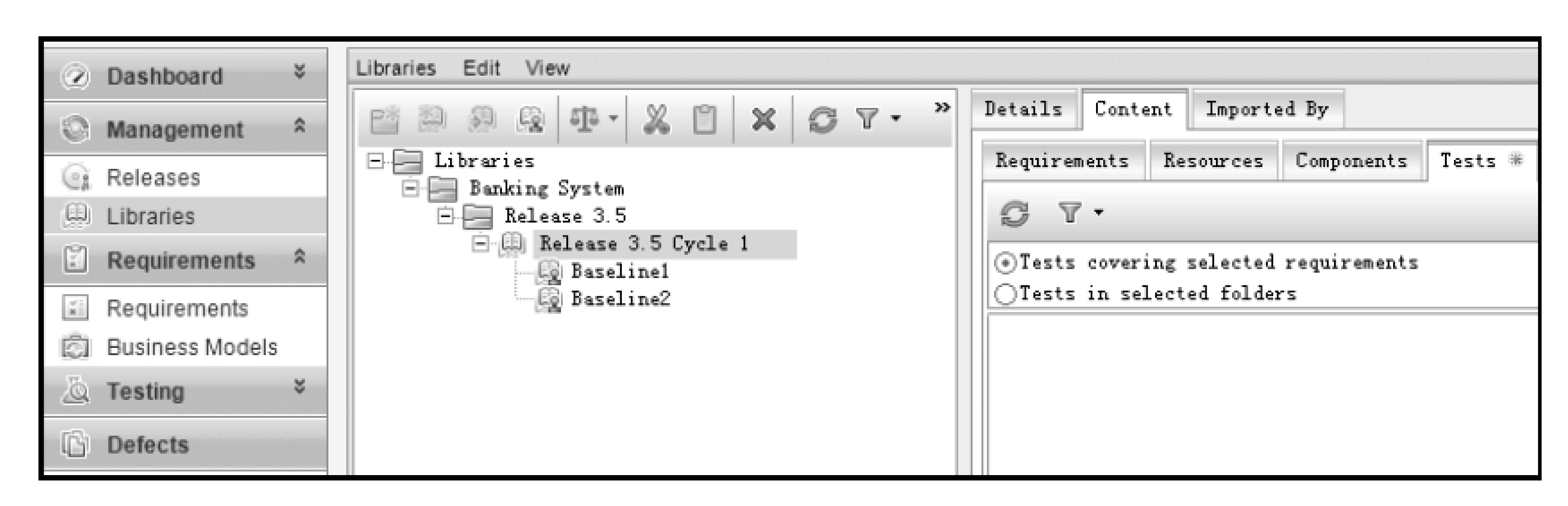


图 20-9　在库中添加覆盖需求测试

20.3　创建基线

基线包含库中所有被定义的实体，包括需求、测试和测试资源。在创建库之后，可以创建基线。基线可以让你随时记录项目的变化。当创建基线时，ALM 首先核实库。连接验证过程将检查库中是否存在与库之外的实体有关联的实体。

20.3.1　基线概述

基线是库在特定时间点的快照。在应用程序开发生命周期中，可以运用基线标识各个重大的里程碑。基线还包括：

1) 库中实体之间的关系，比如追踪能力和覆盖范围。

2) 库中测试运行需要的库之外的所有相关联的实体，比如可以通过调用测试和测试资源基线随时记录项目的变化情况。

可以使用以下方式使用基线：

1) 对比应用程序开发生命周期各个阶段的基线。例如，可以通过对比库中的两个基线来适时评估需求变化所带来的影响。然后可以更新对应项目中相关的测试。

2) 也可以把基线与库中当前的实体进行对比。

3) 固定测试集到基线。这可以确保在运行测试集时，能运行存储在基线中指定的测试版本。

4) 使用基线共享库中的实体。确保库中的实体能在同一项目或其他项目中得以重新使用。要导入库，那么库必须包含基线。

5) 通过对比两个基线查看库在不同开发阶段的变化。

6) 绑定测试集到基线，这可以确保当运行测试集时，ALM 能运行存储在基线中的指定

的测试版本。

20.3.2　创建基线

步骤如下：

(1) 在 Libraries 树中，选择一个库。

(2) 单击 Create Baseline 按钮，如图 20-10 所示创建基线，开始验证过程。

a) 在 Baseline Name 框中输入新基线的名称。

b) 单击 OK。基线被添加到 Libraries 树中，创建过程开始。基线的创建将在后台执行，这个过程会花费一定的时间。在基线创建过程中可以继续 ALM 的相关工作。

(3) 在 Details 选项卡中单击 View Log 按钮。

a) 弹出 Log:Create Baseline 对话框并显示进程。

b) 单击 Close 以关闭对话框。

c) 单击工具条的 Refresh 按钮以刷新显示。基线的详细信息显示在 Details 标签中。在 Details 选项卡中，单击 Description 面板添加对基线的描述。

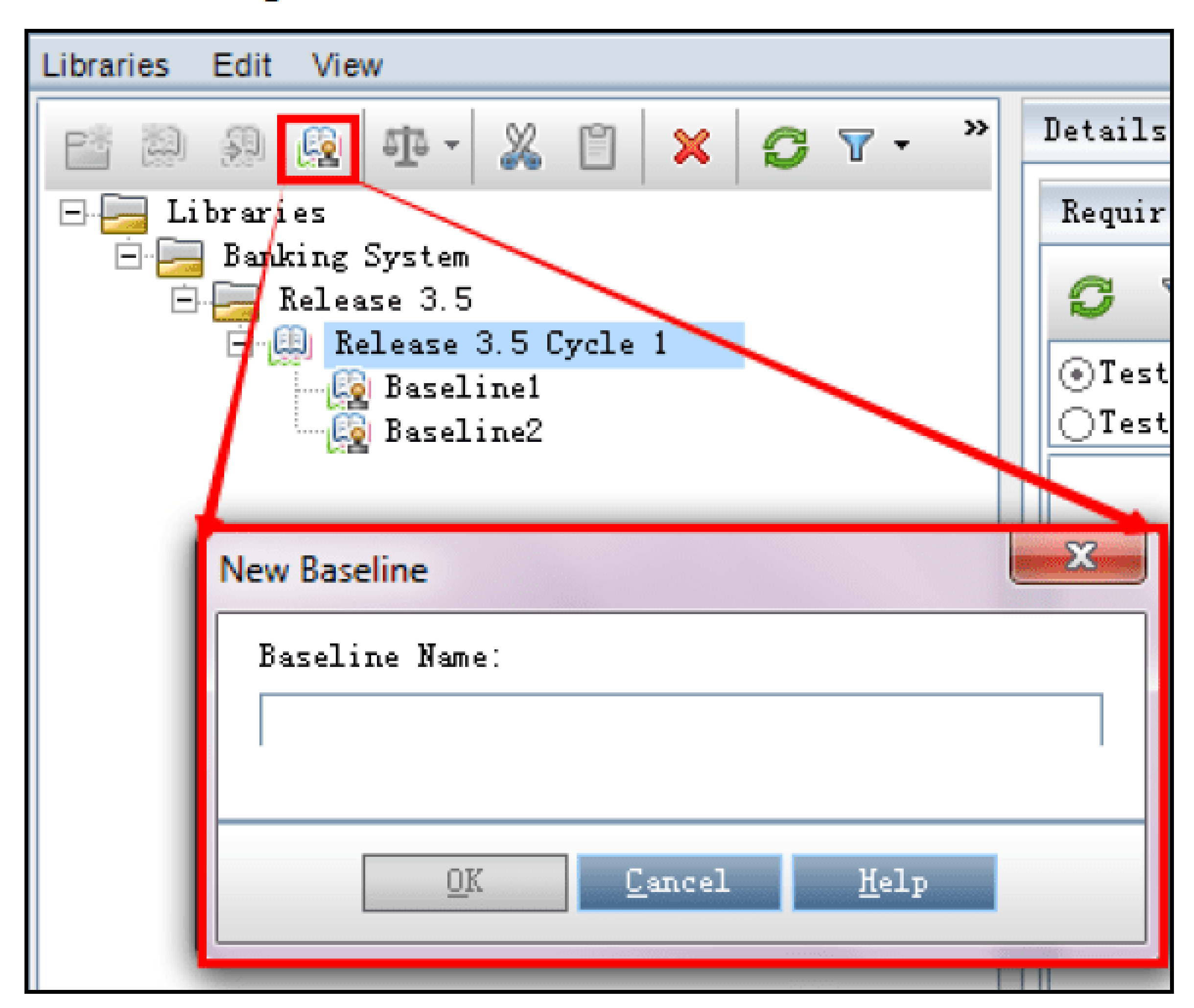


图 20-10　创建基线的过程

表 20-1 列出了创建库和基线时涉及的相应字段。

表 20-1　字段描述

字　段	描　述
名称	Libraries 文件夹、库或基线的名称
创建人	创建库或基线的用户的名称
创建日期	库和基线的创建日期

(续表)

字　段	描　述
Library ID	ALM 自动分配给库的唯一数字 ID
Baseline ID	ALM 自动分配给基线的唯一数字 ID
修改时间	上一次重命名基线或修改描述信息的日期和时间
描述	Libraries 文件夹、库或基线的描述信息

注意：

1) 可以添加用户自定义字段并修改 Details 选项卡中任何字段的标注信息。

2) 可以用 Script Editor 限制和动态改变 Libraries 模块中的字段和值。

3) 可以对比库中的两个基线。例如，通过对比不同开发阶段的基线来评估项目中需求变化带来的影响。然后更新相应项目中相关的测试。

4) 可以将基线和当前库中的实体进行对比。例如，假设在新发布开始时创建一个基线。随着时间的推移，库中的需求会发生改变。要确定产品开发是否按照计划运行，可以将初始基线时的需求和当前库中的需求进行对比。

5) 当确定一个实体是否被修改过时，ALM 不考虑对 Target Release 和 Target Cycle 的修改。

20.4　基线对比工具

20.4.1　基线对比

步骤如下：

(1) 弹出 Compare Baselines Tool 对话框。

(2) 在 Libraries 树中，展开一个库并选择一个基线。单击 Compare To 按钮或者选择 Libraries | Compare To，然后选择要进行对比的基线(如图 20-11 所示)。

(3) 选择基线，将选择的基线和库中的另一个基线对比。弹出 Select Baseline 对话框。单击 Browse 按钮并从列表中选择一个基线，单击 OK。

(4) 单击 OK，关闭 Select Baseline 对话框。

(5) 选择 Current Entities 以将选择的基线和库中的实体进行对比，如图 20-12 所示。

基线在单独窗口中显示，窗口中显示了最近创建的基线，右侧窗口显示了当前的实体。在每个窗口中，库中的实体都以在特定模块中定义的同一层次结构显示。Changes 字段标识了两个基线之间的区别。

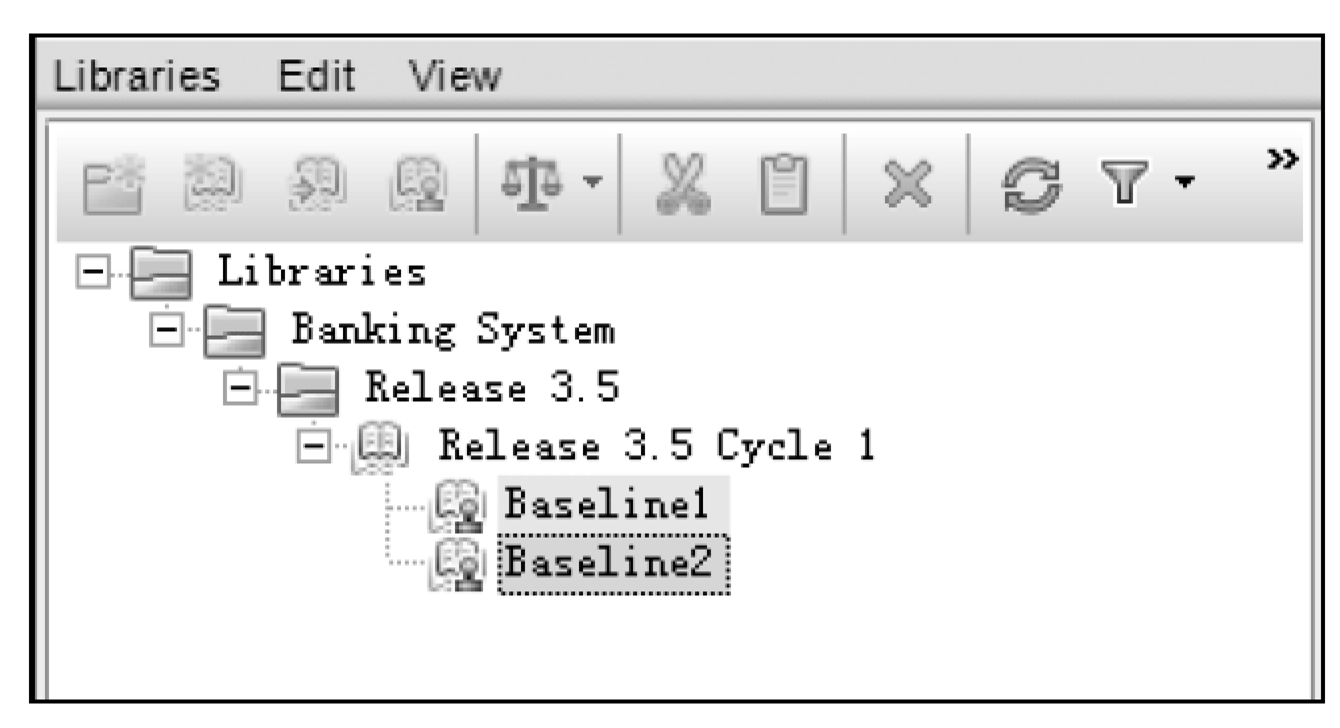


图 20-11　基线对比

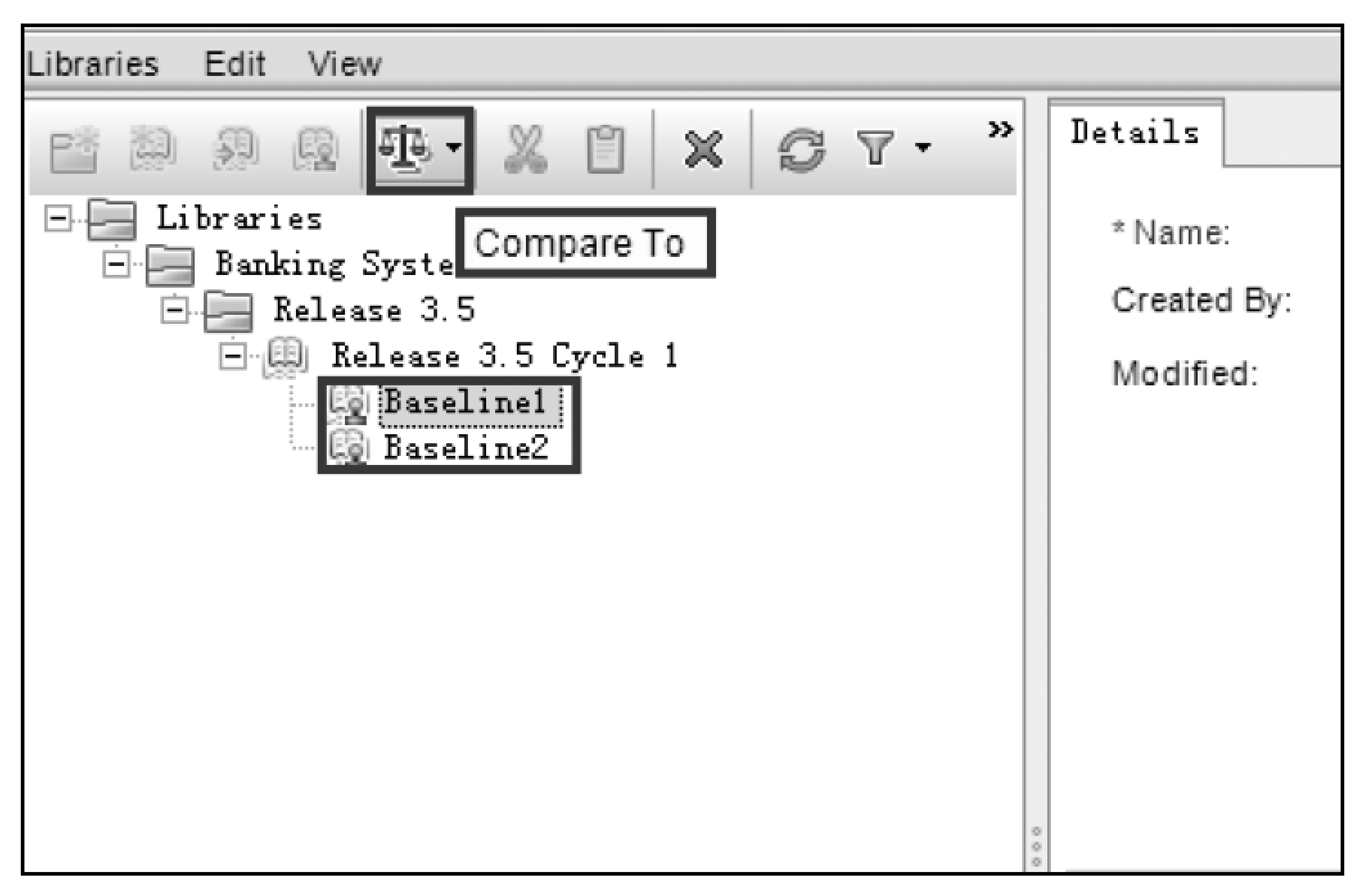


图 20-12　基线对比的步骤

可以通过 Compare Baselines Tools 来识别以下几种类型的变化，如表 20-4 所示。

- 添加、修改、删除或移除的实体的个数在各自窗口上方的计数器中标识出来。
- 被删除或移除的实体显示在树的原始位置，以位置标识符标识。这种持续的层次结构可以让你对比基线之间的变化。

表 20-2　基线变化列表

变　化	描　述
添加	实体不存在于前期基线中
删除	实体不存在于新的基线中
修改	实体和基线之间存在区别
移除	在新的基线中，实体在树结构中的位置发生变化

(1) 单击右侧窗口中的 Go To Next Change 按钮，跳转到下一个变化，或者单击 Go To Previous Change 按钮，跳转到前一个变化，如图 20-13 所示。

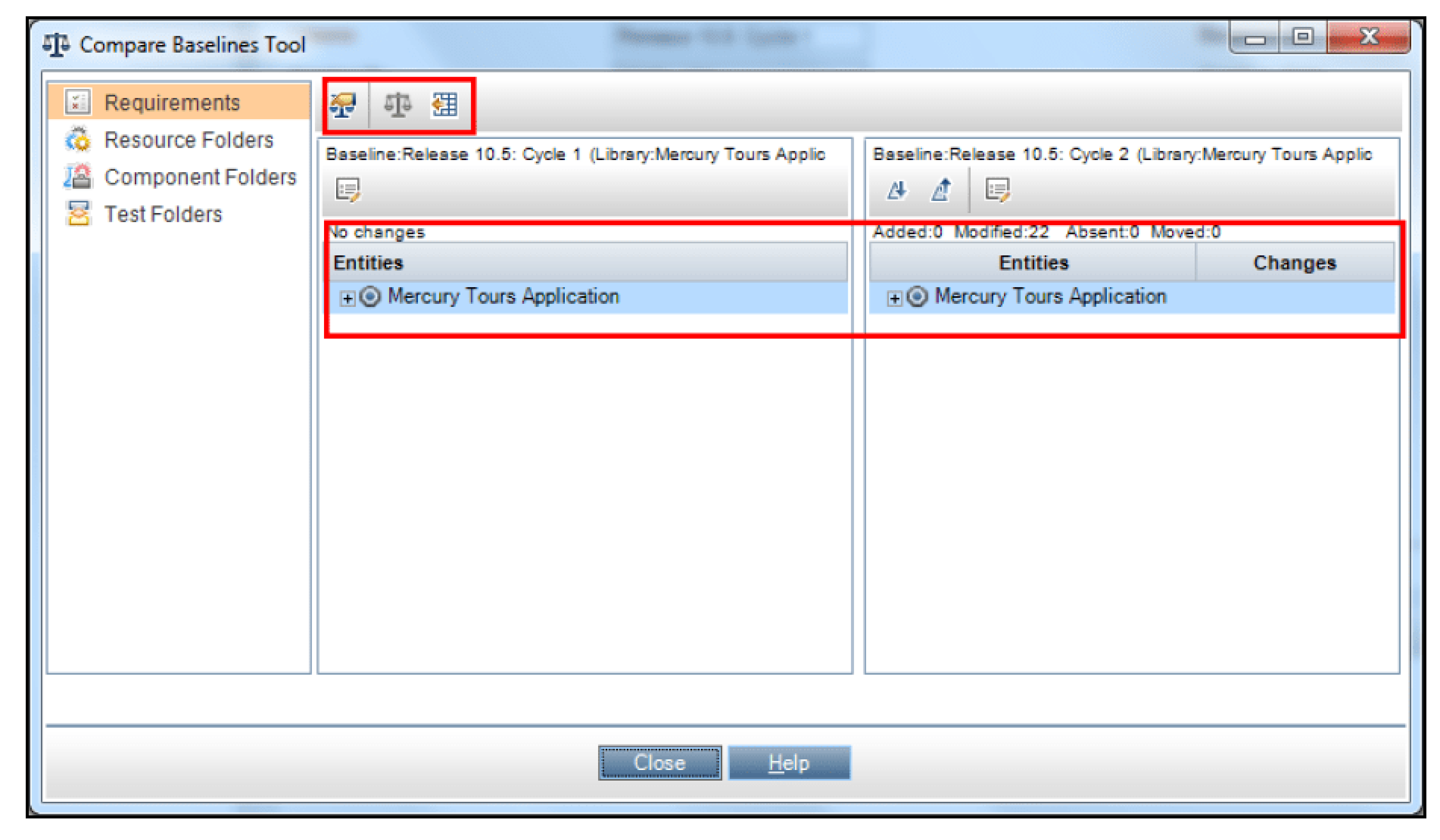


图 20-13　基线变化显示

(2) 看一个实体的详细信息，选择实体并在工具条处单击 Details 按钮。弹出实体的 Details 对话框。例如，选择一项需求并单击 Details 按钮，Requirements Details 对话框将被打开，对话框中显示了所选需求在所选基线中的详细信息。

(3) 比较不同基线中被修改的实体，选择实体并在工具条处单击 Compare Entities 按钮，如图 20-14 所示。

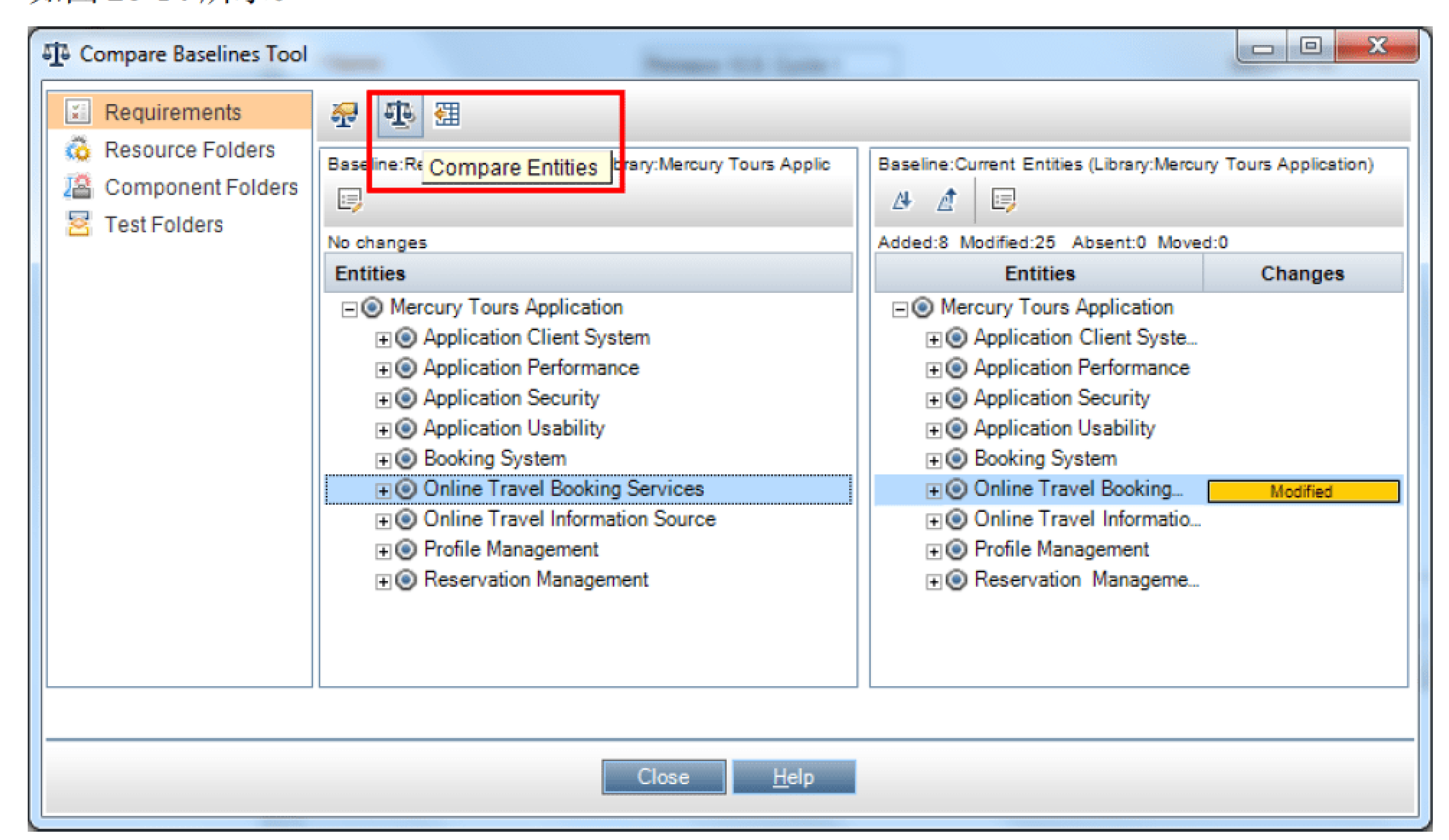


图 20-14　基线添加显示

(4) 弹出 Compare Entities 对话框，其中显示了每个基线中实体的详细信息。

(5) 使用 Compare Baselines Tools 设置选择项，单击工具栏中的 Comparison Setting 按钮。

获得更多的信息，可查看 Configuring the Comparison Settings。

20.4.2　实体对比

具体步骤如下：

(1) 在 Compare Baselines Tool 或 Compare Library Tool 对话框中，选择一个被修改的实体并单击 Compare Entities 按钮。弹出 Compare Entities 对话框，如图 20-15 所示。

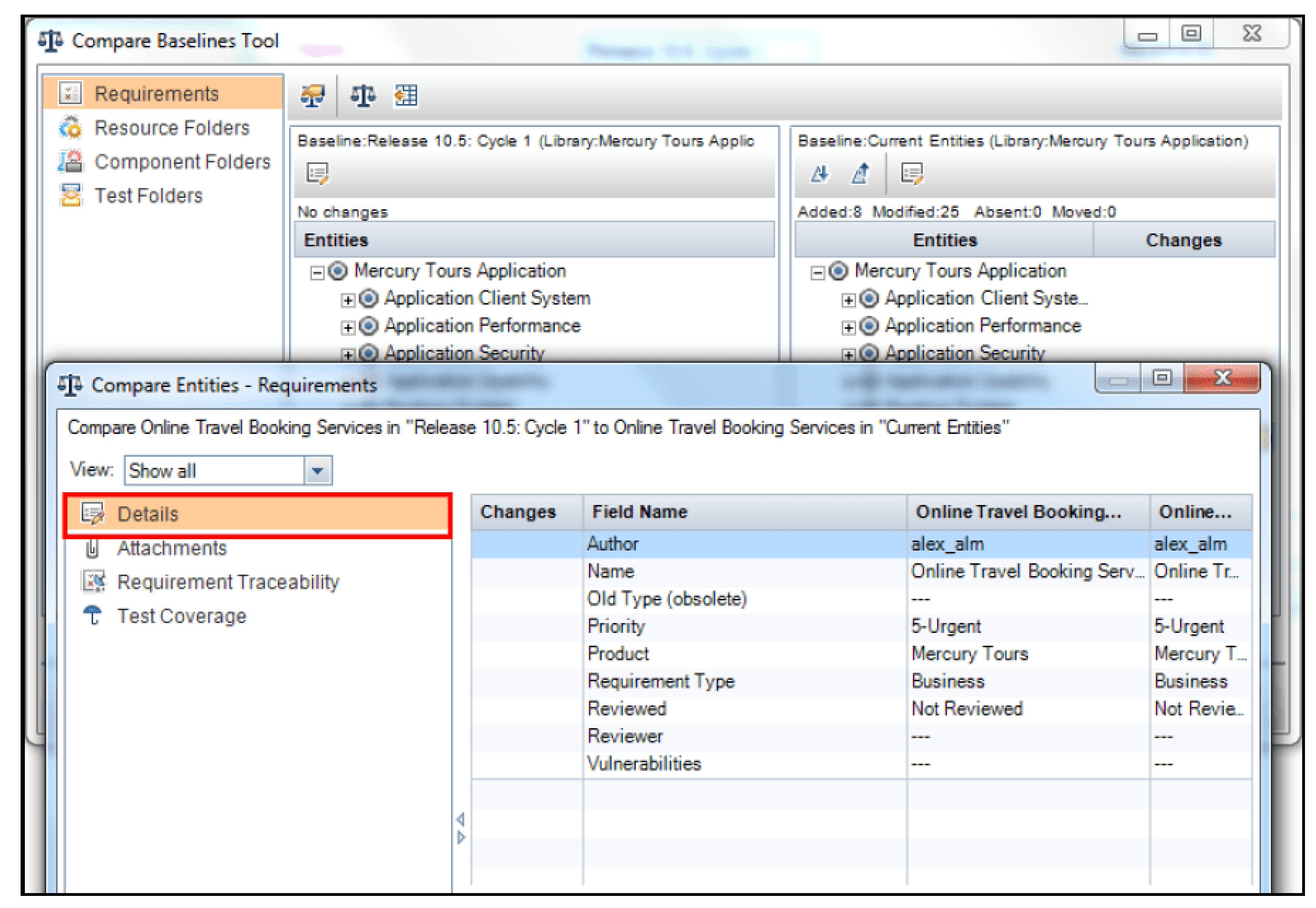


图 20-15　实体对比方法

(2) 在 Compare Entities 对话框中，在侧边栏单击 Details，Field Name 列显示实体的字段。可以对比不同实体版本的某一字段值。被修改过的字段会在 Changes 列中被标记为 Modified。

(3) 版本控制：在版本控制之下，一些字段的变化不会被存储。

(4) 在 View 框中选择以下一项：

1) Show All:显示所有实体的字段

2) Show Only Changed：显示字段值发生变化的字段。

3) Show Only Unchanged：显示字段值未发生变化的字段。

(5) 单击 Rich Text 标签，对比实体的富文本。Rich Text 选项卡中的每个窗口都显示了相关实体版本的富文本是否被添加、删除或修改过。

(6) 在侧边栏单击任何按钮，查看对比结果，例如跟踪能力和覆盖范围。这些按钮显示的内容取决于进行对比的实体的类型。

(7) 在侧边栏单击 Attachments，对比实体的附件。Changes 列显示了附件是否被添加、删除或修改过。

(8) 可以通过打开或保存附件来查看并对比变化。

1) 要查看附件，双击要查看的附件或选择附件并单击 Open Attachment 按钮。

2) 要保存附件，选择附件并单击 Save As 按钮。在 Save As 对话框中，为文件命名并单击 Save，附件被下载并保存到指定的位置。

20.5　配　置

可以为对比库或基线定义设置。Comparison Settings 对话框使你能够在确定一个实体是否有被修改过时选择 ALM 考虑的字段。

注意，在确定一个实体是否被修改过时，ALM 不考虑 Target Release 和 Target Cycle 字段的变化。

配置对比设置的步骤如下(如图 20-16 所示)：

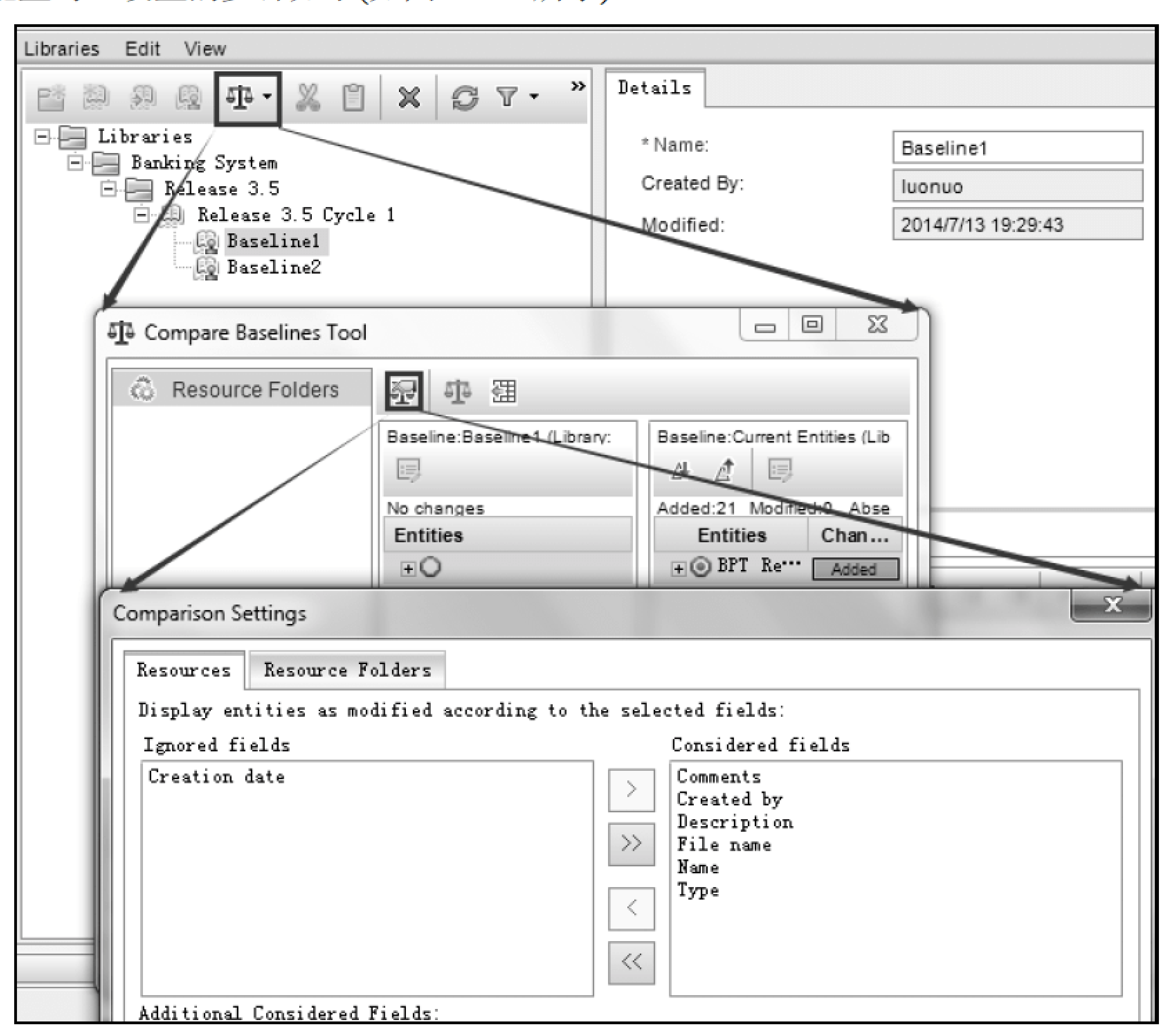


图 20-16　配置对比设置方法

(1) 在 Compare Baselines 或 Compare Baselines Tools 对话框中，单击 Comparison Settings 按钮，弹出 Comparison Settings 对话框。

(2) 单击计划配置的实体类型选项卡。例如，单击 Tests 选项卡，弹出当前设置。

(3) Considered Field:ALM 显示。

(4) Ignored Field:ALM 不显示。

(5) 选择字段名并单击单箭头按钮(<和>)，在 Ignored Fields 窗口和 Considered Fields 窗口

之间移动它们。单击双箭头按钮(>>和<<)，将所有的实体从一个列表移到另一个列表。也可以双击一个字段名，在不同列表之间移动字段。

(6) 在 Additional Considered 字段下，选择任意一个字段包含进来。为了让 ALM 知道要忽略覆盖范围，必须清空需求和测试标签中的 Coverage Checkbox。单击 OK 按钮来关闭 Comparison Settings 对话框。单击 Compare Baselines Tool 或 Compare Libraries Tool 中的 Close 按钮。

20.6　修改 Libraries 树

可查看存储在基线里的实体的所有版本，包含 Requirements、Business Components、Test Plan 和 Test Resource 模块中的每个实体。也可以对比两个不同的基线版本。

查看基线/版本历史记录：在树和网格中选择一个实体，单击 History 选项卡。

- Baseline: 在 Versions And Baselines 选项卡中，在下拉菜单中选择 View By 的值为 Baselines。
- Version: 在 Versions And Baselines 选项卡中，在 View By 窗口中选择 Versions。

在选定基线的 Description 下，查看用户在创建基线时输入的描述信息。

对于 Versions，显示了以下表格中的列：

- 要对比两个版本，按住 Ctrl 键，选择要进行对比的版本并单击 Compare 按钮。Compare Entities <Entity>对话框出现并显示了 Field Name、两个版本的字段值和一个标识变化类型的 Changes 列。
- 当基线属于同一个库时可以对两个基线进行对比。要对比两个基线，按住 Ctrl 键，选择要进行对比的基线并单击 Compare 按钮。

20.7　库元素

(1) 重命名 Libraries 文件夹、库和基线。

(2) 从 Libraries 树中选择一项。选择 Edit|Rename，或者右击要重命名的项，选择 Rename。编辑名称并按 Enter 键。移动 Library Folders 和 Libraries 可以将 Library 文件夹和库移到 Libraries 树的不同位置。移动一个 Library Folder 时，它的库和基线也将被移动。移动库时，它的基线也会移动。所以不能移动基线和 Libraries 根文件夹。

小技巧：可以将 Library Folder 或 Library 拖到 Libraries 树中一个新的位置，除 Library 文件夹、Library 或 Baseline 外。

(3) 不能将基线的库与导入的或是自动生成的库进行对比。从固定的测试集中删除一个基线以清除它。可以删除 Library 文件夹、Library 和 Baseline。但是不能删除包含 Libraries 的 Library 文件夹和包含基线的 Library。在项目中删除一个 Library 或 Baseline 而不要删除

Library 的实体。

(4) 在 Libraries 树中选择 Library 文件夹或选择一个 Library，移动多项时，按住 Ctrl 键并选择要移动的项。

(5) 单击 Cut 按钮，或者选择 Edit|Cut。

(6) 在 Libraries 树中选择一个文件夹。

(7) 单击 Paste 按钮，或者选择 Edit|Paste。

(8) 删除 Library 文件夹、库或基线。

(9) 在删除基线之前考虑以下因素：

1) 在 Libraries 树中选择一项。如要删除多项，按住 Ctrl 键并选择要删除的项。

2) 单击 Delete 按钮。或者选择 Edit|Delete，弹出配置信息。

3) 单击 Yes 以确认要删除的项。

20.8　为基线绑定测试集

确定为基线的测试集只能包含也存在于基线中的测试。当为基线确定测试集时，不属于基线组成部分的测试将从测试集中移除。此外，所有运行的测试也从测试集中删除。

确定基线的测试集：

(1) 在 Test Lab Module 中，从测试集树中选择一个测试集。

(2) 选择 Tests Sets | Pin to Baseline，或者右击并选择 Pin To Baseline。

(3) 弹出 Select Baselines 对话框并显示 Libraries 树。

(4) 展开 Libraries 树，选择一个基线并单击 OK。弹出配置确认信息，单击 Yes 确认(如果测试集包含的测试不存在于 Test Plan 模块中，ALM 将从测试集中删除这些测试。此外，所有运行于这个测试集中的测试被删除)。

(5) 要删除一个确定的测试集，选择此测试集并选择 Tests Sets | Clear Pinned Baseline。弹出配置确认信息，单击 Yes 确认删除。

(6) 可以为基线绑定测试集，把测试集与所指定基线里的测试联系起来。当运行一个为基线绑定的测试集时，ALM 运行指定基线中的测试版本。

Select Baseline 对话框

图 20-18 所示对话框能为基线确定测试集。这把测试集里的测试和所指定的基线中的测试联系起来。当运行一个确定为某一基线的测试集时，ALM 执行指定基线中的测试版本。测试集里的测试与当前 TEST PLAN 模块中的测试有关联。

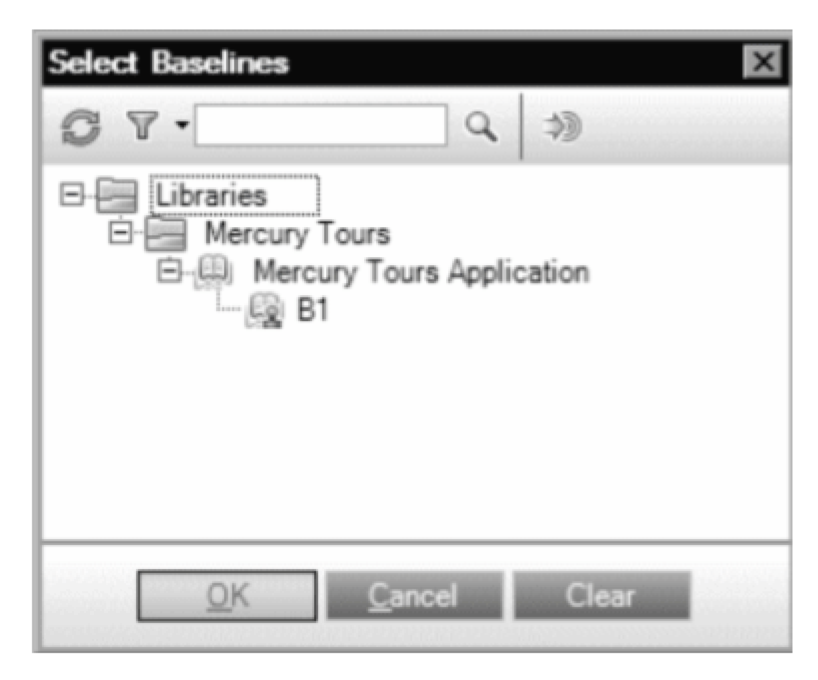


图 20-18 在 Select Baselines 对话框中选择测试集

习题与思考题

1. 本章用到了哪些 Library 管理模块?
2. 什么是基线?
3. 什么是 Libraries 树?
4. 什么是基线的历史记录?

练习：Library 管理

在本练习中，你将打开一个存在的项目。你要执行以下任务：

第 1 部分：检查存在的库

第 2 部分：定义一个库

第 3 部分：定义一个基线

第 4 部分：对比基线

第 5 部分：为基线绑定测试集

(该练习的指导步骤请参见本章正文)

参考文献

[1] (美)Alan Page、Ken Johnston、Bj Rollison，译者：张爽、高博、欧琼、赵勇等. 微软的软件测试之道[M]. 机械工业出版社，2009.

[2] (美) Tim Riley、Adam Goucher，译者：张爽、吕灵、鲍臣礼.测试之美[M]. 机械工程出版社，2010.

[3] 郑文强、马均飞. 软件测试管理[M]. 电子工业出版社，2010.

[4] 张爽、方敏、欧琼等著. 微软 360 度：成功与成长[M]. 电子工业出版社，2008.

[5] 张爽、栾跃、李雨航、王志峰等著. 微软 360 度：企业与文化[M]. 电子工业出版社，2007.

[6] 百度百科. 软件测试管理体系[G/OL]. [2014-7-4].http://baike.baidu.com/view/6296965.htm.

[7] 百度百科. ISO9001[S/OL]. [2014-7-4]. http://baike.baidu.com/view/114518.htm.

[8] 微百科. 建立软件测试体系[M/OL]. [2014-7-1]. http://www.baike.com/wiki/%E8%BD%AF%4%BB%B6%E6%B5%8B%E8%AF%95%E7%AE%A1%E7%90%86%E4%BD%93%E7%B3%BB.

[9] 杨学明. 软件测试管理公开课[EB/OL]. [2013-10-9]. http://www.taoke.com/opencourse/193689.htm.

[10] 中国 IT 实验室. 软件测试管理的基本要素[EB/OL].http://softtest.chinaitlab.com/qtpm/827695.html.

[11] 佚名. 软件测试管理解决方案简介[EB/OL].[2013-6-5].http://www.gwssi.com.cn/products-6-7.html.

[12] 中国 IT 实验室. 软件测试管理平台 V1.0[EB/OL]. [2014-5-2]. http://softtest.chinaitlab.com/qita/744194.html.

[13] 中国 IT 实验室. 测试管理要素分析[EB/OL]. [2013-9-10]. http://softtest.chinaitlab.com/qtpm/897530.html.

[14] 中国 IT 实验室. 不对所有可能性进行测试的原因[EB/OL]. [2014-6-9]. http://softtest.chinaitlab.com/hot/902971.html.

[15] 北大软件. 测试管理系统[EB/OL]. [2014-3-19]. http://www.beidasoft.com/wwwroot/gswz/298/600.shtml.

[16] 火龙果软件. 敏捷测试的方法和实践[EB/OL]. [2013-9-10]. http://www.uml.org.cn/softwareprocess/20111253.asp.

[17] 软件测试网. 敏捷测试人员的十条法则[EB/OL]. [2013-10-29]. http://www.51testing.com/html/15/n-244415.html.

[18] 软件测试网. 测试管理中的问题[EB/OL]. [2013-8-25]. http://www.51testing.com/html/99/n-845999.html.

[19] 佚名. 测试技术-软件测试策略[R/OL]. [2014-3-2]. http://staff.ustc.edu.cn/~shizhu/zlbz/7.pdf.

[20] 中国 IT 实验室. 软件测试的金字塔体系[EB/OL]. []. http://softtest.chinaitlab.com/pm/883980.html.

[21] 百度百科. TMM——软件测试能力成熟度模型[EB/OL]. [2014-7-8]. http://baike.baidu.com/view/5527435.htm.

[22] 八宝玛丽. 软件测试之测试策略[EB/OL].[2014-5-4]. http://www.soft6.com/tech/15/150289.html.

[23] yangdaliang. 测试策略和测试计划的区别.[EB/OL]. [2013-9-5]. http://blog.csdn.net/yangdaliang/article/details/5612425.

[24] 百度文库. 软件测试策略说明书[EB/OL]. [2013-11-1]. http://wenku.baidu.com/view/3a169370a417866fb84a8e44.html.

[25] 成人教育. 需求分析的目标与任务[EB/OL]. [2013-9-5]. http://www.edushanghai.org/Item/5850.aspx.

[26] 黎连业、张晓冬、吕小刚. 软件能力成熟度模型与模型集成基础[M]. 机械工业出版社，2011.

[27] luluping. 软件需求分析方法[EB/OL]. [2013-9-5].http://www.cnblogs.com/luluping/archive/2009/06/18/1505754.html.

[28] 百度文库. 测试需求分析[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/68abe5d026fff705cc170a8f.html

[29] 百度文库. 性能测试需求分析[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/575ffb1da300a6c30c229f3c.htm.

[30] 百度文库.性能测试需求分析[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/200bb3d7b9f3f90f76c61b08.html.

[31] UML 软件工程组织. 高级测试管理的工具和技术[EB/OL]. [2013-9-5]. http://www.uml.org.cn/test/200610185.htm.

[32] developerWorks 中国. Rational 测试解决方案[EB/OL]. [2013-9-5]. http://www.uml.org.cn/test/200610185.htm.

[33] IEEE-CS "IEEE Standard for Software Test Documentation", 35. IEEE-SA Standards Board[S].

[34] shangyichen. 究竟什么是测试需求[EB/OL]. [2013-9-5]. http://blog.csdn.net/shangyichen/article/details/5050513.

[35] (美)莱芬韦尔 等著，蒋慧 译. 软件需求管理用例方法(第二版)[M]. 中国电力出版社，2004.

[36] 中国 IT 实验室.浅析软件项目管理中的需求管理[EB/OL]. [2013-9-5]. http://www.

uml.org.cn/test/200610185.htm.

[37] 软件测试网.一切从需求管理开始[EB/OL]. [2013-9-5]. http://www.51testing.com/html/61/n-824161.html.

[38] ISO 官方网站. ISO 9000 - Quality management[S/OL]. [2013-9-5]. http://www.iso.org/iso/iso_9000.

[39] 中国质检出版社第四编辑室. 计算机软件工程国家标准汇编(第2版)[M]. 中国质检出版社、中国标准出版社，2011.8.

[40] pandana. 软件质量的度量. [EB/OL]. [2013-9-5]. http://blog.csdn.net/pandana/article/details/7307373.

[41] 百度文库.软件质量介绍[EB/OL].[2013-9-5]. http://wenku.baidu.com/view/37ca9af77c1cfad6195fa726.html.

[42] 新浪博客. 软件质量的概念[EB/OL].[2013-9-5]. http://wenku.baidu.com/view/37ca9af77 c1cfad6195fa726.html.

[43] 百度文库. 软件质量度量教程[EB/OL].[2013-9-5]. http://wenku.baidu.com/view/f4ea411714791711cc7917ff.html.

[44] 百度百科. 软件质量因素[EB/OL]. [2013-9-5]. http://baike.baidu.com/view/551050.htm.

[45] 百度文库. ISO9001 八项质量管理原则[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/91b1416548d7c1c708a145dc.html.

[46] 百度文库. 软件质量管理实践[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/ecab2459312b3169a451a486.html.

[47] 百度文库. 软件质量属性[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/14f4e5eee009581b6bd9ebbd.html.

[48] 百度文库. 软件质量管理与控制[EB/OL]. [2013-9-5]. http://wenku.baidu.com/view/dd7e6c4d2b160b4e767fcf27.html.

[49] 陈绍英. 软件测试与持续质量改进(第3版)[M]. 人民邮电出版社，2011-7.

[50] 中国工业产品质量. PDCA 循环—戴明环[EB/OL]. [2013-6-7]. http://www.quality.org.cn/glff/qmzlgl/2557.shtml.

[51] 互动百科. 戴明环[EB/OL]. [2013-9-4]. http://www.baike.com/wiki/%E6%88%B4%E6%98%8E%E7%8E%AF.

[52] Lencioni P. The Five dysfunctions of a Team: A Leadership Fable[M]. Random House Audio, 2006-4-4.

[53] Harvey Robbins、Michael Finley. The New Why Teams Don't Work [M]. Berrett-Koehler、Revisededition, 2001-1-1.

[54] HP. HP ALM11.5 用户指导文档[EB/OL]. [2013.7.1]